N° d'ordre : 2292 Année 2005

THÈSE

Préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

En vue de l'obtention du
Doctorat de l'Institut National Polytechnique de Toulouse

Spécialité
Systèmes Informatiques

par

Anis YOUSSEF

Réseau de communication à haut niveau d'intégrité pour des systèmes de commande-contrôle critiques intégrant des nappes de microsystèmes

---

Soutenue le 22 novembre 2005 devant le jury :

Président	**Christian FRABOUL**
Rapporteurs	**Françoise SIMONOT-LION** **Thierry DIVOUX**
Examinateurs	**Jean-Jacques AUBERT**
Directeurs de Thèse	**Yves CROUZET** **Agnan de BONNEVAL**

A la mémoire de Ommi Fatma.

A Houda, à qui aucun mot ne peut décrire ma gratitude.

A mes parents Mounira et Abdelamjid
pour leur inépuisable soutien
Merci …

A Ommi Kmar pour sa gentillesse
et ses prières.

A mon frère Karrouma pour
toute la bonne humeur
et la joie qu'il apporte.

AVANT-PROPOS

Les travaux présentés dans ce mémoire ont été menés au Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS).

Je tiens à remercier Monsieur Jean-Claude LAPRIE et Monsieur Malik GHALLAB, Directeurs de recherche au CNRS, qui ont successivement assuré la direction du LAAS-CNRS depuis mon arrivée, de m'avoir permis d'accomplir mes travaux dans ce laboratoire.

Je remercie également Messieurs David POWELL et Jean ARLAT, Directeurs de Recherche au CNRS, responsables successifs du groupe de recherche Tolérance aux fautes et Sûreté de Fonctionnement informatique (TSF), pour m'avoir permis de réaliser ces travaux dans ce groupe.

Je tiens à remercier Messieurs Jean Jacques AUBERT, Ingénieur de Recherche, Patrice BROT et Pascal TRAVERSE d'AIRBUS France, pour leurs conseils très constructifs, leurs remarques pertinentes et leurs apports techniques qui ont contribué au bon déroulement des travaux de ma thèse qui s'inscrit dans le cadre d'un contrat de collaboration de recherche entre le LAAS-CNRS et AIRBUS France. Ce fut une expérience extrêmement enrichissante. J'ai énormément apprécié le travail en équipe avec les partenaires industriels du projet au niveau scientifique et humain. J'ai tiré de nombreux enseignements de leurs compétences et de leurs expériences.

J'exprime ma profonde reconnaissance à Monsieur Yves CROUZET, Chargé de Recherche au CNRS, et Monsieur Agnan de BONNEVAL, Maître de conférences à l'Université Paul Sabatier de Toulouse, qui ont dirigé mes travaux de thèse, pour m'avoir encadré et soutenu tout au long de cette thèse. Je les remercie pour leurs conseils, leur soutien et leur disponibilité.

Je tiens également à souligner leurs compétences scientifiques dont j'ai beaucoup tiré parti. Nos discussions pendant les longues fins de semaine d'avant soutenance, qui étaient parfois très nocturnes, et leurs apports à la fois techniques et pédagogiques m'ont été d'un grand secours. Leurs lectures attentives des différentes versions du manuscrit, leurs conseils très constructifs et leur soutien dans les moments les plus difficiles ont fortement contribué au bon déroulement des travaux présentés dans ce mémoire.

Qu’ils trouvent ici un témoignage de mon estime et de ma reconnaissance. Je remercie également Monsieur Jean ARLAT, Directeur de Recherche CNRS, pour ses conseils et sa disponibilité. Je suis très honoré d’avoir eu la chance de travailler avec vous.

Je remercie Monsieur Christian FRABOUL, Professeur des Universités à l'Institut National Polytechnique de Toulouse, pour l’honneur qu’il m’a fait en présidant mon jury de thèse.

Je remercie tout particulièrement Madame Françoise SIMONOT-LION, Professeur des Universités à l'École des Mines de Nancy, et Monsieur Thierry DIVOUX, Professeur des Universités à l'Université Henri Poincaré de Nancy, pour avoir accepté la charge d’être rapporteur.

Un grand merci à ma femme Houda pour sa patience, son soutien inépuisable, son aide et ses encouragements dont j'avais fort besoin surtout dans la dernière ligne droite. Qu’elle trouve ici un témoignage de ma reconnaissance.

Dans ces moments importants, je pense très fort à ma famille : ma mère Mounira, mon père Abdelmajid et mon frère Karrouma “Forza l'EST” qui m’ont toujours soutenu et aidé.

Ils n'ont voulu pour rien au monde rater ma soutenance qui était pour mon père “l'un des meilleurs jours de sa vie”. Ils ont fait le déplacement depuis la Tunisie et ils étaient très fiers de ma réussite. Je les remercie du fond du cœur pour tout ce qu'ils me donnent.

Un grand Merci à Haj Slouma “Toutes les marques c'est de la merde, n’achetez que Siemens”, à Olfa Kaddour “Ca ne colle pas, ça s'enlève”, à Allouch “Je ne sais pas pourquoi je ne vais pas bien … kisss…”, à Ayda “Que la paix soit avec vous”, à Taha “Sabah ilkhir ya achiri, ayya tochroub kaoua”, à Afef “On programme quelque chose ce WE ?”, à Elyes “Ana inkassar wa Olfa tlim” et à Meriem “Sayyibni khallini naamel jaou”.

Merci à toute l'équipe TSF pour son accueil, sa bonne humeur et son ambiance scientifique très riche. Je veux remercier Joëlle et Gina, les deux secrétaires pour leur disponibilité et leur gentillesse, l'ensemble des permanents pour leurs critiques et conseils. Et également, l'ensemble des doctorants pour l'ambiance conviviale qu'ils ont su créer : Ana “fromasse”, Eric “Little bouda”, Carlos “Big bouda”, Magnos “Salut juyas”, Nico “Ca va comme un lundi”, Arnaud “La faucheuse”, Guillaume “Mister Tee”, Minh “Chin chao”, Eric “Kangourou rourou”, Vincent Nicomette “Ca va Frimeur”, Marco Killijian, Ludovic Courtès, Benjamin Lussier, Cristina Simache et Christophe Zanon.

Je réserve ici une mention particulière à Tahar Jarboui “Désormais…”, Noureddine Abghour “Yakhi Tharbik ethou”, Anis Sahbani “Ca te dit un match de foot ?”, Ahmed Benjilani “Pizza, ça va”, Moselem Belkhiria “Yiddou fi jibou”, Anis Ziadi “Asfour yganni”, pour nos nombreuses soirées jeux “ghamza, mafia, belotte, …” et pour les barbecues Merguez qui durent jusqu'à l'aube.

Mes remerciements s’adressent également à tous les membres des services *Informatique* et *Instrumentation*, *Documentation–Édition*, *Magasin*, *Entretien*, *Direction–Gestion*, *Réception–Standard* qui m’ont toujours permis de travailler dans d’excellentes conditions.

Au delà du simple contexte professionnel, j’ai trouvé au LAAS un environnement de travail exceptionnellement chaleureux. J’aimerais remercier tous ceux qui ont pu m’aider dans les moments les plus variés.

Table des matières

INTRODUCTION GENERALE

Les importants progrès dans les domaines des réseaux de communications numériques et de la microélectronique ont été à l'origine d'importantes mutations au niveau des systèmes de commande-contrôle. Et d'autres évolutions encore plus importantes sont déjà engagées, induites cette fois par les progrès, tant académiques qu'industriels, dans le domaine des microsystèmes, qui investissent des domaines aussi variés que le médical, les télécommunications, la robotique, l'automobile, l'aéronautique, l'espace, l'environnement, etc. Ainsi, l'introduction de microsystèmes en très grand nombre dans les systèmes de commande-contrôle devrait permettre, par exemple, une commande plus fine en raison du remplacement de quelques actionneurs par une nappe de microactionneurs ayant indubitablement une action plus fine ; ou encore, permettre de répartir les traitements applicatifs. De plus, cette multiplication des actionneurs ou des capteurs est particulièrement intéressante dans le cas des systèmes de commande-contrôle critiques. En effet, cela permettra de considérer différemment la sûreté de fonctionnement de ces systèmes, dans la mesure où, par exemple, il sera plus facile d'admettre de perdre une petite partie des actionneurs ou des capteurs.

Néanmoins, à ce jour, de nombreux défis sont encore à relever, comme en particulier, celui des réseaux de communications nécessaires pour supporter cette très forte intégration de microsystèmes.

Parmi les domaines concernés, l'avionique est un secteur qui, depuis le Concorde, s'est inscrit dans cette dynamique d'évolution des systèmes de commande, avec l'introduction, dans les systèmes de Commandes De Vol (CDV), des systèmes de type *fly by wire*. Et l'utilisation de technologies numériques (au niveau des calculateurs) date du début des années 80, et a été certifiée pour la première fois, par rapport aux normes de l'aviation civile, en 1988 sur l'Airbus A320. Les CDV gèrent les déplacements de l'avion en agissant sur les surfaces de contrôle mobiles de l'avion. Il s'agit donc de systèmes critiques, et sont donc soumis à de très fortes contraintes de sûreté de fonctionnement et réglementés par de sévères normes de certification. En effet, certaines de leurs défaillances, même temporaires, peuvent avoir des conséquences catastrophiques. L'événement redouté, car pouvant amener à un événement catastrophique, est un embarquement d'une surface : surface inutilisable ou dans une position pouvant mettre en danger la structure de l'avion. Le niveau de sûreté de fonctionnement des CDV doit être tel que le taux d'occurrence de l'embarquement d'une surface soit inférieur à 10^{-9}/ heure de vol.

Les évolutions futures pour les CDV sont, à court terme, l'utilisation de liaisons numériques au sein même des systèmes de commandes de vol. Et à plus long terme, il s'agit d'utiliser des nappes de microsystèmes pour, par exemple, remplacer des surfaces de contrôle, tels que la dizaine de *spoiler*s actuels d'un Airbus A320, par des milliers de *microspoilers*, ce qui suppose aussi l'utilisation de liaisons numériques. Ces deux axes d'évolution font partie d'une révolution lente mais inexorable des transports aériens.

Ce contexte a été le cadre initiateur de nos travaux qui visent à définir un système de communication numérique à haut niveau d'intégrité apte à répondre à la fois : 1) aux évolutions à court terme des CDV (tout en assurant un niveau de sûreté de fonctionnement au moins égal à celui des actuels CDV), et 2) également à un des nombreux défis que soulève l'utilisation de nappes de microsystèmes dans des systèmes de commande-contrôle critiques plus généraux.

Dans l'optique de limiter le temps de développement et le coût des solutions envisagées, les orientations que nous avons prises sont, autant que possible, d'une part, d'utiliser des solutions standard au niveau des équipements réseau, et d'autre part, malgré le niveau de criticité des systèmes considérés, d'éviter autant que possible toute forme de duplication des liaisons de communication et des équipements réseau, en tout cas en ce qui concerne la satisfaction de l'intégrité recherchée. Ce dernier point est particulièrement important dans la perspective d'utilisation de nappes de microsystèmes.

Il est important de préciser que, bien qu'une grande partie des travaux présentés dans ce mémoire porte sur la définition d'un système de communication à haut niveau d'intégrité, c'est bien dans le contexte général de l'introduction de nappes de microactionneurs dans les systèmes de commande-contrôle critiques que se situent nos travaux. Il faut également préciser que ces travaux ont été menés en collaboration avec Airbus France dans le cadre d'une convention de recherche et que leur illustration s'appuie fortement sur une application de type CDV. Enfin, bien que ce type application ait assez fortement orienté nos travaux, il n'en n'a pas pour autant limité leur portée. En effet, il s'agit d'une application représentative d'une classe bien plus large de systèmes de commande-contrôle, dont une des spécificités se traduit au niveau de leur système de communication, par le fait que l'intégrité des communications recherchée est à considérer sur un lot de messages et non sur un message seul.

L'ensemble des travaux que nous avons menés peut être appréhendé selon trois volets. Le premier a consisté à réaliser un premier tour d'horizon sur les microsystèmes afin de déterminer les implications, de leur intégration massive dans les systèmes de commande-contrôle, sur le réseau de communication de ces systèmes. Alors, à partir des éléments de base retenus pour un tel réseau, nous avons analysé si les fonctions standard de contrôle d'erreur les plus couramment mises en œuvre au niveau d'un système de communication sont compatibles avec le niveau d'intégrité visé. Le deuxième volet a consisté à proposer de nouvelles fonctions de contrôle d'erreur permettant de faire face aux insuffisances révélées lors du volet précédent et à valider cette proposition. Il a alors été possible, dans un troisième volet de revenir plus spécifiquement sur l'intégration de nappes de microsystèmes dans les systèmes de commande-contrôle. Ce dernier volet a permis de faire le point sur la situation actuelle et de donner des indications sur des adaptations à apporter au niveau du système de communication ainsi que sur les autres travaux encore à mener.

*

* *

Ce mémoire est structuré en cinq chapitres.

Dans le premier chapitre, après une introduction sur les microsystèmes, nous présentons un premier aperçu des avantages que pourrait offrir leur utilisation sous forme de nappes dans des systèmes de commande-contrôle critiques. Nous présentons ensuite les principales caractéristiques des systèmes de commande-contrôle considérés dans nos travaux. Par la suite, après avoir situé nos travaux dans le contexte de la sûreté de fonctionnement, nous décrivons d'abord un ensemble de propriétés qui peuvent permettre de caractériser, d'un point de vue général, une communication intègre, puis nous indiquons les propriétés que nous retenons compte tenu des caractéristiques des systèmes considérés. Nous terminons ce chapitre en présentant notre cas d'étude – les systèmes de commandes de vol – pour lequel, après avoir présenté les évolutions passées et futures, nous donnons les spécifications fonctionnelles et non-fonctionnelles qui y sont rattachées et pertinentes pour les travaux que nous avons menés.

Le deuxième chapitre présente les contraintes et les caractéristiques retenues pour le système de communication envisagé pour les systèmes de commande-contrôle considérés. Pour l'architecture, nous ne donnons que des éléments de base nécessaires aux analyses effectuées par la suite : principalement, une architecture à base de bus, comportant des nœuds intermédiaires (d'interconnexions) actifs, notamment, du fait du nombre élevé de microsystèmes. Puis nous présentons les aspects protocolaires, fondés sur une étude des protocoles standard les plus largement utilisés dans les réseaux locaux. Une analyse de risques est ensuite menée pour identifier les différentes sources d'altération des communications et y associer des modèles d'erreurs. Nous évaluons alors le niveau d'intégrité qu'il est possible d'atteindre avec les fonctions standard de contrôle d'erreur, ceci pour chacune des sources d'altération identifiée, et nous comparons ce niveau avec l'objectif d'intégrité de haut niveau qu'il est nécessaire d'atteindre pour que le taux d'occurrence de l'événement redouté, qui pourrait résulter d'une couverture de détection insuffisante des altérations, soit inférieur au seuil de criticité de 10^{-9}/h.

Dans le troisième chapitre, compte tenu des insuffisances révélées par l'analyse précédente, nous proposons une nouvelle fonction de contrôle d'erreur, qui couvre les différents aspects que nous avons retenus pour une communication intègre, et qui permette d'obtenir le niveau d'intégrité requis pour notre cas d'étude. Au niveau de ce chapitre, nous donnons les principes retenus pour la nouvelle fonction de contrôle, mais pas encore la description de sa mise en œuvre réelle. La priorité de notre démarche à ce niveau est d'aboutir à un haut niveau d'intégrité tout en limitant le coût en termes de bits de contrôle rajoutés. Pour cela, nous prenons en compte une des spécificités de la classe des systèmes de commande-contrôle considérée, et qui est que l'intégrité des communications est à considérer sur un lot de messages et non sur un message seul. Nous montrons ensuite comment la solution proposée permet de faire face à certains modes de défaillance (du système de communication), qui ont été identifiés comme critiques lors de l'analyse rapportée dans le chapitre 2. Nous analysons aussi dans ce chapitre quel peut être l'impact, sur le taux d'occurrence de l'événement redouté, de la stratégie de recouvrement utilisée suite à la détection d'une erreur.

Dans le quatrième chapitre, nous présentons une mise en œuvre particulière de la fonction de contrôle d'erreur proposée, en mettant d'abord en avant les analyses préliminaires montrant la pertinence de ce choix de mise en œuvre. Puis, nous décrivons les travaux de validation effectués sur cette mise en œuvre. Cette validation est faite par rapport aux spécifications du cas d'étude, aussi bien en termes d'intégrité des communications que de respect des contraintes temporelles. Le but étant de montrer l'effet de la nouvelle fonction de contrôle sur le taux d'occurrence de l'embarquement des surfaces de contrôle en le comparant systématiquement au taux de 10^{-9}/h. La validation s'appuie sur des modèles Matlab/Simulink que nous avons développés pour représenter la solution proposée.

Le cinquième et dernier chapitre s'attache à décrire un ensemble de perspectives sur la base d'un état de l'art des microsystèmes. Nous commençons par développer largement l'état de l'art abordé dans le chapitre 1 concernant les microsystèmes individuels, en nappes et en réseaux. Nous y présentons différents exemples d'applications qui font déjà partie de notre vie courante ainsi que des exemples d'applications étudiés dans le domaine plus spécifique de l'avionique. Puis, nous décrivons en quoi l'utilisation de nappes de microactionneurs dans un système de commandes de vol devrait permettre de considérer différemment leur sûreté de fonctionnement. Nous terminons en présentant les autres défis qui sont à relever dans d'autres domaines de compétence, pour que puisse aboutir l'objectif à long terme de remplacer les surfaces de contrôle actuelles d'un avion par des milliers de microsurfaces.

Chapitre 1 - Systèmes de commande-contrôle et nappes de microsystèmes

Parmi les très nombreux défis que soulèvent les microsystèmes, notre intérêt porte sur celui de la définition de systèmes de communication adaptés à l'utilisation massive de microsystèmes par des applications de type *Système de Commande-Contrôle* (*SCC*). Notre cas d'étude est l'application qui est à l'origine de nos travaux : les systèmes de *Commandes De Vol* (*CDV*) dans le domaine de l'aéronautique, qui pourraient avoir à commander des nappes de quelques centaines, milliers, ou plus, de microsurfaces de contrôle. Dans le cadre de cet objectif à long terme, notre préoccupation centrale est de développer un système de communication sûr de fonctionnement, plus particulièrement à haut niveau d'intégrité pour les messages échangés.

Dans ce chapitre, nous commençons donc par donner un premier aperçu, qui sera développé au chapitre 5, des définitions et du spectre des avantages et des défis encore posés par les microsystèmes. Nous montrerons que, dans notre contexte, les aspects que nous prendrons en compte se limitent à considérer des microsystèmes fixes et en très grand nombre. À eux seuls, ces aspects sous-tendent une part importante des travaux présentés dans ce mémoire. Nous définirons alors les systèmes considérés, les SCC, puis la terminologie utilisée pour la sûreté de fonctionnement et préciserons nos considérations sur l'intégrité des communications en général, puis dans le cadre de nos travaux. Enfin, nous présenterons notre cas d'étude, les CDV, et nos hypothèses et spécifications fonctionnelles et non fonctionnelles pour nos travaux.

1.1 Des MEMS aux réseaux de microsystèmes

L'acronyme *MEMS* signifie littéralement *Micro Electro-Mechanical Systems* (Micro Systèmes Électromécaniques) et sous-entend l'intégration de *fonctionnalités hétérogènes* sur une même puce. Les MEMS trouvent leur origine dans le développement des *technologies microélectroniques*, qui ont permis d'exploiter les *propriétés électriques* du silicium pour des mises en œuvre matérielles micrométriques du traitement électrique d'information. Puis, l'idée d'exploiter aussi les *propriétés mécaniques* du silicium a conduit à réaliser sur une puce des fonctions de contrôle et d'actionnement. Il était dès lors possible d'intégrer sur une même puce des microcomposants mécaniques (ex. : capteurs, actionneurs), et d'autres d'électronique de traitement d'informations de contrôle et de commande [Esteve & Campo 2004, Christou 2001].

Aujourd'hui, de nombreux produits sont à maturité industrielle et certains sont déjà fabriqués en masse (par des sociétés comme Bosch ou Motorola) : puces à ADN, commutateurs RF, microbobines pour les télécommunications, etc. Un exemple typique, très largement répandu dans le domaine automobile, est l'accéléromètre pour déclenchement d'airbag (sac gonflable), qui intègre sur une puce de 3 mm^2 un microcapteur et son électronique de traitement [Randell & Muller 2000].

Si les microsystèmes suscitent un tel intérêt en termes de recherche et de marché, c'est que leur caractéristique fondamentale, la *miniaturisation*, leur confère de très significatifs avantages par rapport aux systèmes usuels, tant dans le cadre d'une utilisation individuelle que collective.

Pour définir en quoi les microsystèmes interviennent dans nos travaux, nous présentons donc leurs définitions, un aperçu de leurs avantages et de leurs diversités sur un plan individuel, puis collectif, étant précisé que la plupart de ces points seront repris plus en détail au chapitre 5.

1.1.1 MEMS et microsystèmes individuels

1.1.1.1 Définitions et variantes

Selon les communautés, l'interprétation des critères définissant le concept de MEMS diffère. De plus, ce concept est désigné par différents termes : en Europe, un acronyme utilisé est *MST* (Micro Systems Technologies), au Japon, il est question de *micromachines* [Curtis 1999], et si des éléments optiques interviennent, on parle de MOEMS (Micro Opto-Electro-Mechanical Systems). Enfin, le concept originel de MEMS est aujourd'hui souvent englobé dans celui de *microsystème*. En définitive, sur la base des mots de l'acronyme MEMS, on constate les variations suivantes.

- Pour "*Micro*", tout le monde s'accorde sur le fait que cela traduit une réduction des dimensions du système. Mais, pour certains, il ne s'agit que d'un *facteur relatif* de réduction, sans changement fondamental de technologie (ex. : microsatellite de quelques dizaines de kg), alors que pour d'autres, il s'agit d'un *facteur absolu* de réduction lié à l'utilisation de technologies micrométriques de fabrication.
- "*Electro-mechanical*" traduit l'association de micro composants électroniques et mécaniques. Mais aujourd'hui, ces derniers sont bien souvent la traduction de la mise en œuvre de principes d'autres domaines : optique, thermique, etc. Et, en définitive, d'autres types de composants entrent dans la constitution des MEMS.
- "*Systems*" traduit pour certains, seulement un simple capteur ou actionneur ou autre composant élémentaire, alors que pour d'autres, il s'agit d'un assemblage de composants élémentaires distincts, hétérogènes et coopérants. Cet assemblage doit présenter des capacités de traitement, d'interaction avec l'environnement, voire de communication. Soit donc, au moins un ensemble de capteurs et/ou d'actionneurs (pour différentes natures de variables à mesurer ou contrôler) et une unité de traitement. Soit en fait, les composants assurant les fonctions nécessaires à l'autonomie de fonctionnement d'un système. À l'échelle micrométrique, cela signifie l'intégration de ces fonctions sur une même puce pour former des MEMS "intelligents" [Coumar 2003] (cf. figure 1.1).

Les distinctions faites dans le point précédent font que, pour certains, *MEMS* et *microsystèmes* recouvrent la même chose, alors que d'autres distinguent le composant élémentaire (le MEMS) et l'assemblage de MEMS (le microsystème).

Aujourd'hui, dans les faits, le terme microsystème désigne encore bien souvent un assemblage de composants absolus et relatifs, ou parfois même, un assemblage de composants uniquement relatifs, simplement réduits en surface ou en volume. Mais à terme, l'objectif est bien d'intégrer les fonctions usuelles des systèmes classiques, pour réaliser des microsystèmes absolus. Dit autrement, assembler des MEMS élémentaires pour reconstituer au niveau micrométrique le monde des machines macroscopiques [Judy & Motta 2002].

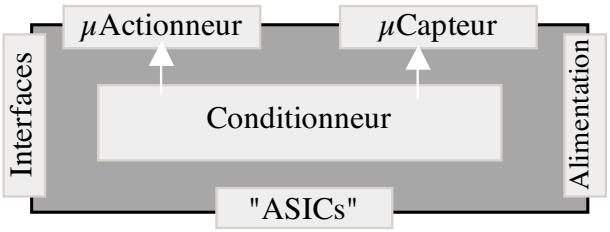


Figure 1.1 - Composition d'un MEMS

Pour notre part, dans le cadre de nos travaux, nous utiliserons le terme microsystèmes pour désigner des systèmes réduits au sens large, même si nous donnons une priorité au cas d'un facteur absolu de réduction.

1.1.1.2 Avantages

Tout l'intérêt des microsystèmes vient de cette miniaturisation qui apporte des avantages avérés et attendus très importants et ouvre de nouvelles perspectives par rapport aux systèmes usuels, malgré l'existence de limitations. Ces avantages sont d'autant plus significatifs que le facteur de réduction est grand, et que le domaine d'application visé est celui des systèmes temps réel ou embarqués.

Miniaturiser un système, c'est d'abord réduire son poids et son volume (son encombrement), et donc ses besoins énergétiques. Mais c'est aussi permettre de *déporter ou localiser "au plus près"* des moyens d'observations, d'actionnements, de traitements et de correction. Ainsi, il est possible d'introduire des systèmes dans de nouveaux *espaces physiques* (ex. : le corps humain). Mais il est aussi possible d'accéder à un *niveau bien plus fin de granularité* des informations (ex. : détection de vibrations sismiques inférieures au millionième de la gravitation terrestre), ou encore des non-linéarités de certaines grandeurs physiques (ex. : écoulement d'air sur une aile d'avion). Dit autrement, les microsystèmes sont sensibles et réactifs à des microvariations, inexploitables jusqu'ici, tout en étant bien moins intrusif vis-à-vis de l'environnement.

Par ailleurs, miniaturiser à un niveau micrométrique permet un très haut niveau d'intégration sur une même puce de fonctionnalités hétérogènes. Cela simplifie la conception et la fabrication de systèmes qui sont aussi plus performants et plus fiables. Notamment, parce que la longueur et le nombre d'interconnexions sont moindres, ce qui diminue les parasites et les durées de transfert des informations entre les différentes fonctions. De plus, ces éléments sont, en particulier, un des facteurs d'accroissement de la réactivité des systèmes.

Enfin, la miniaturisation réduit le coût de revient global, notamment au niveau de leur fabrication, de par les types et quantités de matières premières, et de par les possibilités de production de masse avec des procédés de fabrication éprouvés avec des retours d'expérience significatifs.

1.1.1.3 Limites

Mais les microsystèmes présentent aussi des limitations. En premier lieu, il y a les limitations intrinsèques liées à la miniaturisation et en particulier à la taille, la sensibilité et les problèmes d'énergie. La très petite taille rend très difficile ou impossible *l'accessibilité* pour des vérifications ou des réparations, et augmente *la fragilité physique à l'environnement* (ex. : chocs, vibrations). Elle rend aussi les microsystèmes *inaptes à agir sur un environnement macroscopique*. Ainsi, des modes d'actionnement spécifiques aux microsystèmes ne produisent des forces que de quelques nano Newtons. Par ailleurs, être sensible à des microvariations signifie être également *sensible à des micro-perturbations* (souvent sans effet sur les macrosystèmes), comme des imperfections structurales des matériaux ou des radiations. Ce qui vient entacher la *fiabilité* et les *performances*. Enfin, il ne faut pas oublier les *problèmes énergétiques*, et ce quels que soient les progrès réalisés dans ce domaine.

En second lieu, il y a les limitations liées au manque de maturité, qui en réalité, traduisent les nombreux défis encore à relever : 1) au niveau *technologique*, pour arriver à mieux maîtriser le haut niveau d'intégration, les diverses technologies de fabrication, sans parler des problèmes de production, stockage et consommation d'énergie ; 2) au niveau de la *conception*, où il faut en particulier développer des outils spécifiques [Tanner 2000] ; 3) au niveau *sociétal*, pour faire accepter au grand public l'utilisation massive des microsystèmes dans la vie de tous les jours.

1.1.1.4 Diversités

Aujourd'hui, la panoplie et la maturité des technologies de microfabrication, avec notamment l'augmentation continuelle du degré d'intégration, conduit à une très grande diversité des microsystèmes sur plusieurs plans : structurels et fonctionnels, disciplines scientifiques et technologies impliquées dans la conception, domaines d'application et secteurs économiques.

En effet, le type, le nombre et la complexité des fonctions d'un microsystème ne cessent de croître et de se diversifier. Et la conception d'un microsystème donné met désormais toujours en jeu une pluralité de disciplines scientifiques et des technologies associées. Car si les microdispositifs électroniques et mécaniques sont toujours les constituants de base d'un microsystème, il s'avère que, d'une part, les microdispositifs mécaniques traduisent souvent la mise en œuvre de principes d'autres disciplines, et d'autre part, d'autres types de microdispositifs sont aujourd'hui également utilisés [Bhansali 2001]. En considérant le spectre des domaines d'application et des secteurs économiques, presque toutes les disciplines sont impliquées : électronique, mécanique, physique, optique, thermique, fluidique, chimie, biologie, etc.

Les microsystèmes trouvent désormais leur utilité dans de nombreux *domaines d'application*. Pour ne citer que quelques exemples : *instrumentation* (ex. : capteurs de navigation, de pression, de température), *périphériques* (ex. : stockage de données, têtes d'impression), *communications* (ex. : connecteurs, multiplexeurs, composants radiofréquences), *commandes de procédés*. De ce fait, les microsystèmes investissent la majorité des *secteurs économiques* [Esteve & Campo 2004] : automobile, aérospatial, communications, robotique, médical, domotique, environnement, défense. Et même si aujourd'hui ne sont bien répandus que des produits assez simples pour des marchés à grande diffusion (automobile ou téléphonie mobile), l'emploi massif de produits bien plus complexes est une réalité très proche.

1.1.2 Réseaux de microsystèmes

Il est naturellement venu de vouloir exploiter à grande échelle les avantages des microsystèmes individuels par une utilisation collective [Bohringer & Donald 1998], c'est-à-dire d'une collection ou d'un ensemble de plusieurs microsystèmes (jusqu'à des dizaines de milliers). De plus, ce principe même introduit de nouveaux avantages, mais aussi de nouvelles problématiques. En effet, à ce jour, cette exploitation à grande échelle est bien moins mature que celle des microsystèmes individuels. Et ce, même si l'on peut citer de nombreux exemples, comme les *nappes de capteurs de présence ou de suivi* pour des applications domotiques ou de surveillance médicale (ex. : détection de chutes, gestion thermique de l'habitat), les *matrices de micropropulseurs* pour micro et nano satellites, ou les *matrices de microéjecteurs* pour déposer des gouttes de quelques nano litres. Et plus particulièrement dans l'aéronautique, des réseaux de microsystèmes sont prévus pour surveiller les efforts sur la structure de l'avion, détecter un feu dans les réacteurs, ou encore modifier les propriétés de la couche limite en aérodynamique.

1.1.2.1 Définitions d'un réseau de microsystèmes

Le fondement d'un ensemble de microsystèmes est que chacun d'eux doit être capable de communiquer avec d'autres systèmes ou microsystèmes. Sinon l'utilisation de cet ensemble est sans intérêt. Mais la nature et la complexité des besoins et des solutions de communication varient selon plusieurs paramètres, dont le nombre, l'hétérogénéité, la mobilité, l'autonomie et les capacités de traitement des microsystèmes. Un ensemble de microsystèmes peut être composé de microsystèmes identiques, fixes et à gestion centralisée sur un calculateur (ex. : pour remplacer un capteur ou un actionneur classique). Mais à l'autre extrême, cet ensemble peut aussi être composé de microsystèmes hétérogènes, mobiles, coopérants et ne communiquant qu'entre eux.

La diversité des cas de figures en termes de constitution et d'organisation fait, qu'en première approche, ces ensembles de microsystèmes sont très souvent désignés par le terme générique de "*réseau"* de microsystèmes. Plus finement, les termes *matrices*, *nappes* et *nuages* sont aussi utilisés (cf. § 5.2.1), souvent pour désigner des communications moins sophistiquées, que celles généralement sous-entendues par le terme *réseau*, dans les réseaux usuels.

À ce stade, nous ne détaillerons pas toutes ces distinctions. Pour notre part, dans la suite, pour simplifier la rédaction, nous utiliserons les termes *réseaux* et *nappes*, en précisant néanmoins que par ces termes, nous distinguerons en général le cas de communications *entre* les microsystèmes eux-mêmes, du cas de communications *avec* des microsystèmes de l'ensemble.

Enfin, ajoutons qu'une classification assez usitée [Collet *et al.* 2004] distingue : 1) les *réseaux de microcapteurs,* qui permettent au moins la collecte de données, 2) les *réseaux de microactionneurs,* qui permettent de distribuer ou de répartir des traitements et des actions, 3) les *réseaux de microcapteurs et microactionneurs* qui héritent des propriétés des deux classes précédentes avec en plus des possibilités d'asservissements locaux très rapides.

1.1.2.2 Nouveaux avantages

Le principe même d'utiliser une "*collection"* de microsystèmes procure des avantages supplémentaires pour les applications s'appuyant sur de tels ensembles. D'abord, la complexité et le type des applications envisageables sont étendus, avec la possibilité de distribuer des traitements, ou simplement, d'une part, d'étendre et d'affiner la collecte de données, et d'autre part, de combiner des actions microscopiques pour élaborer des actions macroscopiques.

Par ailleurs, la souplesse et la fiabilité de ces applications sont accrues sur certains aspects, du fait qu'il y ait davantage de configurations ou de modes dégradés possibles. En effet, d'une part, utiliser plusieurs microsystèmes permet d'introduire à moindre coût de la redondance et de la diversification pour limiter les conséquences d'une défaillance d'abonnés du réseau, sachant que, d'autre part, le niveau d'autorité de chaque microsystème étant plus faible, sa défaillance porte moins à conséquence et sera davantage tolérable. De plus, dans le cas d'interactions entre des microsystèmes du même ensemble, il est aussi possible de développer de l'auto-adaptabilité localisée, donc plus rapide, tant en fonctionnement normal qu'en cas de défaillance d'un ou plusieurs microsystèmes, défaillance qui pourrait être détectée, voire compensée, par un ou des voisins.

Enfin, des gains en coûts sont attendus. Mais à ce jour, leur appréciation est plus difficile que pour les microsystèmes individuels. Car, par exemple, selon la structure physique de la nappe, le remplacement d'un abonné peut nécessiter en fait le remplacement de toute la nappe.

1.1.2.3 Nouvelles problématiques

Le principe même d'utiliser une "*collection*" de microsystèmes introduit aussi, et surtout, de nouvelles problématiques par rapport aux microsystèmes individuels. Nous distinguons :

1. les problèmes spécifiques réseau, ou plus généralement de communication,
2. les problèmes des applicatifs s'appuyant sur ces réseaux.

Au niveau applicatif, il s'agit de savoir comment, tout en tenant compte des contraintes spécifiques des microsystèmes, tirer profit au mieux de leurs avantages spécifiques et de leur utilisation collective, comme par exemple, l'exploitation d'un niveau de granularité jamais encore atteint, c'est-à-dire, en particulier des non-linéarités. De plus, à ce jour, les applications se limitent encore essentiellement à une centralisation du traitement et ne s'appliquent qu'à des ensembles limités de microsystèmes. Or, au-delà d'un certain nombre de microsystèmes, une solution centralisée n'est plus possible, de par la fréquence de multiplexage nécessaire pour adresser tous les abonnés. Donc, la problématique de la distribution du traitement reste entière, et il est inévitable de définir des algorithmes et des mécanismes de répartition et de coopération avec, et entre, les microsystèmes pour réaliser des fonctions globales (cf. § 5.2.2.2) [Collet *et al.* 2004].

Répondre à ces problématiques applicatives passe par le développement de réseaux de communication adaptés, dont l'accroissement et la complexification sont inéluctables pour les grands ensembles de microsystèmes. Or, aujourd'hui, ces communications sont encore très réduites et se résument très souvent à des liaisons directes avec un calculateur central.

Les problématiques de base des communications sont donc elles aussi entières. Car, même s'il s'agit des problématiques classiques des réseaux, c'est-à-dire, des problèmes de topologie, d'architecture et de protocoles de communication (diffusion, point à point, routage, identification, temps de latence, etc.), les spécificités des microsystèmes nécessitent d'adapter les protocoles connus ou même d'en développer de nouveaux, notamment de par le grand nombre d'abonnés et de leur forte exigence de consommer très peu d'énergie.

1.1.3 Les microsystèmes dans nos travaux : premières hypothèses

Des sections précédentes, qui ont mis en lumière l'ampleur et la diversité des intérêts d'exploiter des microsystèmes et des défis encore à relever, il vient naturellement que nos travaux ne concernent qu'une petite partie de tous ces défis.

Nos hypothèses de travail découlent du cas d'étude à l'origine de nos travaux, qui sera présenté au § 1.5, où nous compléterons et préciserons la première série d'hypothèses que nous donnons ici pour délimiter en quoi les microsystèmes interviennent dans nos travaux.

Nous nous intéressons à l'exploitation de grands ensembles de microsystèmes (quelques centaines ou milliers) par des applications de commande-contrôle (développées au § 1.2). Dans ce contexte, la problématique visée est celle des réseaux de communication servant de support à ces applications. Plus précisément, nous cherchons à déterminer l'impact, des besoins de commander de tels ensembles, sur les choix architecturaux et protocolaires pour mettre en place un système de communication adéquat en termes d'intégrité des communications (développé au § 1.4).

Parmi tous les paramètres possibles, nous ne prendrons en compte que les contraintes d'un nombre élevé de microsystèmes, en précisant qu'ils sont : en nombre et localisation invariants, sans problème d'énergie, et fonctionnellement peu complexes, au sens qu'ils n'auront pas ou peu de capacités de mémorisation ou de traitement local [Cui & Zhou 1996]. Par contre, ils auront nécessairement des capacités de communication, même si pour l'instant, nous ne traiterons pas le cas de communication entre les microsystèmes eux-mêmes. De ce fait, dans nos travaux, nous avons choisi d'utiliser le terme de *nappe* plutôt que de *réseau*.

Enfin, précisons clairement que nous intervenons en tant *qu'utilisateurs* de microsystèmes, et non en tant que concepteurs. Nous ne traitons donc pas des problématiques liées à la conception, à la technologie (y compris celle de l'activation), à la génération et consommation d'énergie ou aux techniques de fabrication des microsystèmes.

1.2 Systèmes de commande-contrôle considérés

Les systèmes que nous avons considérés sont des *Systèmes de Commande-Contrôle* (*SCC*) ***répartis***, ***temps réel*** et ***critiques***, tels que ceux du domaine automobile ou aéronautique par exemple. Il convient donc de décrire les caractéristiques de ces systèmes.

Du caractère ***réparti***, nous retiendrons qu'il se traduit par *l'utilisation d'un réseau de communication*, plus ou moins complexe, indispensable pour répondre à l'augmentation de la distribution et du nombre de calculateurs, de capteurs et d'actionneurs à mettre en relation. Le caractère ***temps réel***, quant à lui, signifie que pour assurer correctement son service, le système doit respecter des contraintes de temps (ponctualité et réactivité) finies et régies par la *dynamique* de son environnement. Nous préciserons dans cette partie, des nuances associées au caractère temps réel (du fait des nuances possibles de cette dynamique), et des conséquences sur l'environnement du non respect des contraintes temps réel. Enfin, le caractère ***critique*** étant un des fils conducteurs de nos travaux, il nécessite à lui seul une section, avant de pouvoir présenter les principales caractéristiques des SCC considérés : type de commande et dynamique des systèmes commandés.

1.2.1 Systèmes critiques : sévérité, criticité et certification

Un système est déclaré ***critique*** si une de ses défaillances peut conduire à un ***événement jugé catastrophique*** sur le plan humain, économique ou environnemental. On parle alors de ***défaillance catastrophique***, définie dans [Laprie *et al.* 1996] comme étant telle que "*ses conséquences sont incommensurablement différentes du bénéfice procuré par le service délivré en l'absence de défaillance*". L'intérêt de cette définition est de couvrir la diversité de points de vue sur le caractère *catastrophique*, en ramenant son appréciation à l'appréciation de la "distance" entre le service nominal et celui rendu en cas de défaillance.

Mais cette définition abstraite ne peut pas être utilisée directement : elle est à détailler par rapport au système auquel on veut l'appliquer. Donc, avant de préciser le niveau de criticité des systèmes visés par nos travaux, il convient d'abord de définir la notion de *sévérité* des défaillances d'un système.

La ***sévérité*** ou la ***gravité*** des défaillances est déterminée à partir du classement des conséquences des défaillances sur l'environnement du système (dans notre cas, il s'agit du procédé piloté par le système de commande-contrôle et ses utilisateurs). Les modes de défaillance sont donc ordonnés en niveaux de sévérité, auxquels sont généralement associés des probabilités maximales d'occurrence admissibles. Le nombre, la dénomination et la définition des niveaux de défaillances et des probabilités associées, dépendent du domaine d'application. Ainsi, les sévérités considérées sont au nombre de :

- 4 dans l'aéronautique civile : mineure, majeure, dangereuse, catastrophique,
- 3 dans la production d'énergie nucléaire : incidents mineurs, incidents, accidents graves,
- 4 pour les lanceurs spatiaux : significatif, majeur, grave, catastrophique,
- 4 dans l'automobile : mineure, majeure, grave, catastrophique,
- 4 dans le ferroviaire : pas de dénomination à notre connaissance.

Le tableau 1.1, qui est extrait du guide de la sûreté des systèmes de l'administration fédérale de l'aviation américaine [FAA 2000], décrit le cas de l'aviation civile.

Dénomination des défaillances	*Conséquences des défaillances*	*Probabilité d'occurrence par heure*	
Mineure	Réduction non significative de la sécurité de l'avion. Peut inclure : légère réduction des marges de sécurité ou des fonctions, léger accroissement de la charge de travail, quelques inconvénients pour les passagers.	Probable	$> 10^{-5}$
Majeure	Réduction significative des marges de sécurité ou des fonctions, ou augmentation significative de la charge de travail de l'équipage, ou de l'inconfort des occupants avec possibilité de blessures.	Rare	comprise entre 10^{-5} et 10^{-7}
Dangereuse	Grande réduction des marges de sécurité ou des fonctions, ou détresse physique ou grande surcharge de travail, ou blessure fatale ou sérieuse pour un relativement petit nombre de passagers.	Extrêmement rare	comprise entre 10^{-7} et 10^{-9}
Catastrophique	Empêche la continuité de la sécurité à l'atterrissage ou en vol: l'avion, les passagers et l'équipage sont perdus.	Extrêmement improbable	$< 10^{-9}$

Tableau 1.1 - Niveaux de sévérité dans l'aviation civile

Le niveau de ***criticité*** d'un système se déduira simplement du plus fort niveau de sévérité de ses modes défaillances. Là encore, les dénominations et le nombre de ces niveaux de criticités dépendent du domaine. Par exemple, ils sont notés, par ordre décroissant : de A à C dans le nucléaire, de A à D dans l'aéronautique civile, et de SIL 4 à SIL 1 dans le ferroviaire.

Les classes et les dénominations sont celles établies par des ***règlements de certification*** normalisés, qui traduisent un processus complexe induisant un surcoût limitant les innovations, mais qui sont une garantie pour les utilisateurs et l'environnement des systèmes critiques. En effet, le caractère critique induit d'importantes contraintes supplémentaires en termes de sûreté de fonctionnement lors de la conception et du développement de tels systèmes. La définition et le respect de ces contraintes ne sont pas laissés à la seule charge du concepteur. Un organisme tiers établit des règlements pour traduire les exigences des autorités de tutelle du domaine du système en question. Puis cet organisme vérifie et valide (ou invalide), d'une part, que les règlements soient bien appliqués, et d'autre part, que le niveau imposé de sûreté de fonctionnement soit atteint pour le système. À titre d'exemples, citons comme normes de certification la DO-178B pour l'aéronautique, la NF EN 50128 pour le ferroviaire, l'ECSS pour le spatial et la CEI 61508 pour l'automobile.

Au final, pour un système critique, l'objectif principal est d'assurer que le taux d'occurrence d'un événement catastrophique reste inférieur au seuil fixé pour ce système.

Pour traiter cet objectif, il faut préciser plus avant les caractéristiques du type de système considéré, dont en premier lieu : le type de commande et la dynamique du système commandé.

Mais avant cela, il nous paraît important d'évoquer certains des moyens d'atteindre cet objectif, qui sont communs à tout type de SCC critique, dès lors que leur seuil de criticité est élevé. Dans ce cas, il est généralement fait appel à des architectures à haut niveau de redondance, voire de diversification, matérielle et ou logicielle. Cette redondance peut, par exemple, se traduire par une architecture duplex pour les calculateurs, ce qui permet en cas de détection d'une anomalie de geler les organes de commande correspondants. Un autre exemple est l'utilisation, pour assurer la continuité de la commande, de plusieurs voies de commande, une seule voie étant active à un moment donné. La détection d'une anomalie sur la voie active provoque le basculement automatique sur une autre voie qui prend le relais, étant entendu que la durée de ce basculement doit être compatible avec les contraintes temps réel imposées. Bien entendu, ces moyens sont à affiner selon la catégorie de système considéré.

1.2.2 Type de commande et dynamique des systèmes commandés

1.2.2.1 Commande à état ou à événement

La nature des commandes découle de la manière dont on veut ou doit commander un système. Nous distinguerons deux types de systèmes selon que les commandes véhiculent ce que nous appellerons respectivement des *valeurs absolues* ou des *valeurs relatives (ou différentielles)* par rapport à une référence. Dans le second cas, la valeur de la commande est un incrément (une variation) à appliquer à une grandeur (ou un basculement d'une grandeur booléenne) par rapport à la valeur précédente de cette grandeur (qui sert de référence) : par exemple, se déplacer de 5 m ou tourner de 5°. Alors que dans le premier cas, la valeur de la commande est définie par rapport à une référence commune à toutes les valeurs d'une même grandeur : par exemple, se déplacer pour atteindre la position 5 m, ou tourner pour atteindre 5°.

Lorsque les commandes transitent par un réseau de communication, ces deux notions conduisent respectivement à celles de commande par ***messages d'état*** ou par ***messages d'événement*** [Kopetz 1998]. Pour les SCC critiques, l'utilisation d'une commande par messages d'état, lorsque le SCC le permet, est souvent préférable à celle par messages d'événement, dans la mesure où elle est moins sensible à la perte de messages ou à un ordonnancement incorrect des commandes reçues.

Dans nos travaux, nous faisons l'hypothèse d'envois de commandes par des messages d'état.

1.2.2.2 Dynamique des systèmes commandés

La *dynamique du système commandé* influe aussi fortement sur la façon de le commander, sur sa robustesse naturelle à des défauts de commande, et donc sur les moyens de sûreté de fonctionnement à mettre en place. Là encore, nous distinguons deux classes de systèmes.

Le premier cas est celui des systèmes que nous qualifierons "*à dynamique instantanée*", parce que la durée d'une évolution significative des grandeurs commandées est très proche de celle d'un cycle de commande-contrôle, ce qui ne permet l'envoi que d'une seule commande pendant cette durée. De ce fait, la prise en compte par le système commandé d'une seule commande erronée pourrait avoir des conséquences désastreuses.

Nous opposons ces systèmes à ceux que nous qualifierons "*à dynamique lente*", parce que la durée d'une évolution significative des grandeurs commandées est suffisamment grande par rapport à celle d'un cycle de commande-contrôle, pour permettre d'envoyer, si on le souhaite, plusieurs commandes pendant cette durée. Cela permet de rendre le système commandé pratiquement insensible à la prise en compte d'un certain nombre (jusqu'à un seuil fixé pour l'application considérée) de commandes erronées avant d'atteindre l'événement catastrophique. C'est par exemple le cas de la commande des actionneurs hydrauliques de certaines surfaces de contrôle sur un avion : ils sont dimensionnés pour que le déplacement (de l'ordre de 50°/s) des surfaces ne soit pas trop rapide pour ne pas engendrer d'efforts excessifs sur la structure de l'avion. Précisons qu'il ne peut s'agir que *de commandes à état*, sinon les valeurs successives de commandes se cumuleraient. Ou alors, si des commandes à événement sont utilisées, alors il ne peut y avoir qu'un seul envoi de commande pendant cette durée.

Comme nos préoccupations portent sur la sûreté de fonctionnement, nous ne discuterons pas davantage des conditions conduisant à distinguer ces deux classes de systèmes, comme en particulier le choix, ou la nécessité, du type de commande pour le système à commander.

Il nous suffit de savoir qu'en termes de sûreté de fonctionnement, la distinction entre ces deux classes de systèmes se caractérise par le fait que l'on cherche à se protéger avec un niveau de confiance donné, dans le premier cas, de *chaque occurrence de défaillance*, alors que dans le second cas, on cherche à se protéger d'un *lot d'occurrence de défaillances*, ce qui revient à dire que la protection vise à réagir "suffisamment tôt" et non instantanément. Cette différence est importante dans la manière de traiter certains aspects de la sûreté de fonctionnement.

Notons également, que la caractéristique de dynamique lente rend difficile l'analyse de l'impact des défaillances du système de contrôle sur le système commandé, ce qui fait dire aux auteurs de [Jumel *et al.* 2003] : "il est difficile de lier explicitement les défaillances du système de contrôle à l'apparition d'une défaillance sur le système physique, la plupart des systèmes régulés (système physique avec son système de contrôle) ayant une capacité intrinsèque à corriger des défaillances non permanentes du système de contrôle".

1.2.3 Bilan

La figure 1.2 rappelle que nous cherchons à définir et évaluer des moyens architecturaux et protocolaires pour assurer la sûreté de fonctionnement, en termes plus particulièrement d'intégrité, du réseau de communication d'un SCC commandant des nappes de microsystèmes.

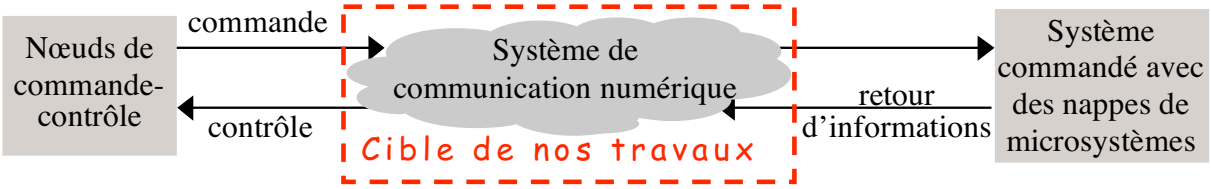


Figure 1.2 - Système de communication numérique d'un SCC

Sachant que la sûreté de fonctionnement pour un système critique a pour objectif principal d'assurer un taux d'occurrence d'un événement catastrophique inférieur à un seuil donné, nos travaux ciblent délibérément les SCC très critiques. C'est-à-dire avec *un seuil fixé à 10^{-9} /h*, ce qui correspond à un niveau *A* de criticité dans l'aviation civile, comme c'est le cas du système réel, support à nos travaux : les commandes de vol électriques (cf. § 1.5). Cette valeur du taux d'occurrence de l'événement catastrophique est aussi couramment retenue pour des travaux dans le domaine de l'automobile, même s'il n'est pas encore dicté par les organismes de régulation correspondants [Wilwert *et al.* 2003].

Par ailleurs, nous avons plus spécifiquement considéré le contexte des *systèmes à dynamique lente* (même si certains résultats semblent pouvoir s'appliquer à des systèmes à dynamique instantanée). Leur caractéristique qui fait que l'on cherche à se protéger d'un *lot d'occurrence de défaillances* influe beaucoup sur ce qui sera défini comme un événement catastrophique au niveau du SCC, et par voie de conséquence, au niveau de son réseau de communication.

De manière générique, la répercussion de cette caractéristique sur le réseau peut se formuler ainsi : c'est la ***répétition de la non-protection*** vis-à-vis des altérations des messages échangés entre les nœuds du réseau qui présente un risque. Nous montrerons au chapitre 3, comment nous exploiterons cela pour proposer une solution pour améliorer le niveau de protection [Youssef *et al.* 2005].

La sûreté de fonctionnement étant un vaste domaine multiculturel et multifacette, il convient de le détailler pour préciser le positionnement de nos travaux sur cet aspect.

1.3 Sûreté de fonctionnement

Les principaux concepts et définitions de base sur lesquels nous nous appuierons par la suite sont ceux établis dans [Laprie *et al.* 1996] et [Laprie 2004]. La sûreté de fonctionnement d'un système est définie comme *"la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans le **service** qu'il leur délivre"*, ou encore, plus synthétiquement, *"l'aptitude d'un système à délivrer un service de confiance justifiée"*, ce qui traduit une relation de *"dépendance acceptée"*. Le *service* définit le comportement du système tel que perçu par ses utilisateurs (autres systèmes, matériels, logiciels ou encore humains, interagissant avec ledit système), étant précisé qu'un système peut être décomposé en sous-systèmes, devenant chacun à son tour un nouveau système. Sur cette base, la sûreté de fonctionnement est structurée en attributs, entraves et moyens.

1.3.1 Attributs

Les ***attributs*** sont les propriétés à satisfaire ou satisfaites par le système, et sont classifiés en :

- ***disponibilité*** : aptitude à être prêt à rendre le service,
- ***fiabilité*** : aptitude à assurer la continuité du service,
- ***sécurité-innocuité*** : aptitude à ne pas engendrer de conséquences catastrophiques pour l'environnement,
- ***intégrité*** : aptitude à ne pas subir d'altérations **inappropriées** d'informations du système,
- ***confidentialité*** : aptitude à ne pas conduire à des divulgations **non autorisées** d'informations gérées par le système,
- ***maintenabilité*** : aptitude à être réparé et à évoluer pour intégrer de nouvelles fonctionnalités.

Il s'agit là des attributs de base qui, bien entendu, sont en partie complémentaires et dépendants les uns des autres. En particulier, ***l'intégrité*** **est un pré requis pour la** ***sécurité-innocuité***, comme pour la *fiabilité* et la *disponibilité*. Ces attributs sont aussi combinables pour définir des *attributs composites*, comme principalement la sécurité-confidentialité qui associe à la confidentialité, la disponibilité et l'intégrité vis-à-vis des actions autorisées. Et bien souvent, on cherche un compromis sur un sous-ensemble de ces attributs, sachant que l'importance accordée à chaque attribut et le niveau qu'il doit atteindre varient selon le système.

1.3.2 Entraves

Les ***entraves*** à la sûreté de fonctionnement traduisent ce qui limite ou affecte les attributs. Elles sont classifiées en *fautes*, *erreurs* et *défaillances :*

- une *défaillance* (D) survient lorsque le comportement du système considéré dévie de sa fonction, autrement dit, le service rendu diverge du service attendu ;
- une *erreur* (E) est la partie de l'état du système susceptible d'entraîner une défaillance ;
- une *faute* (F) est la cause supposée ou adjugée de l'erreur.

Ces trois notions présentent une relation de dépendance de type "cause à effet", qui de plus est relative, c'est-à-dire se déclinant de manière récursive, puisque la défaillance d'un sous-système 1 du système peut à son tour être perçue comme une faute par un sous-système 2 en aval ou par le système qui l'englobe, comme le montre la figure 1.3.

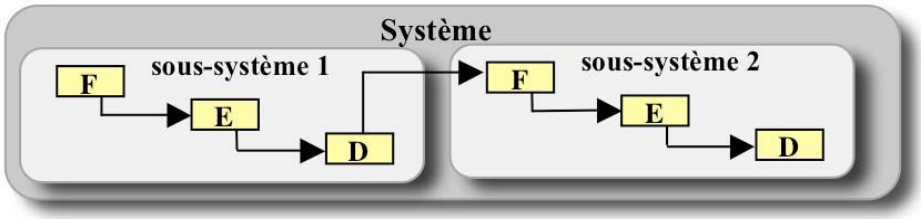


Figure 1.3 - Entraves à la sûreté de fonctionnement

Chacune de ces classes d'entraves est décomposable en sous-classes selon différents points de vue, dont nous ne donnons ici que les grandes lignes pour les fautes et les défaillances.

Pour les *fautes*, il existe huit points de vue : étape d'occurrence, frontières du système, cause phénoménologique, dimension, intention, objectif, capacité et persistance. Il en découle seize classes de fautes élémentaires, sur la base desquelles sont définies une trentaine de *fautes combinées* résultant de combinaisons pertinentes de plusieurs points de vue. Les principales classes de fautes élémentaires sont les fautes :

- de *développement* ou *opérationnelles*, en termes *d'occurrence dans la vie du système*,
- *internes* ou *externes,* en termes de localisation par rapport aux frontières du système,
- *physiques* ou *dues à l'homme*, en termes de *cause phénoménologique,*
- *délibérées* ou *non*, en termes *d'intention*,
- *malveillantes* ou *non*, en termes *d'objectif*,
- *accidentelles* ou *d'incompétences*, en termes de *capacité*,
- *permanentes* ou *temporaires* (liées à des conditions ponctuelles, donc à durée limitée), en termes de *persistance*.

Pour les *défaillances*, selon les points de vue, on distingue comme modes de défaillances :

- les défaillances *signalées* ou *non*, en termes de *domaines de défaillances*,
- les défaillances *en valeur* ou *temporelles*, en termes de *détectabilité*,
- les défaillances *cohérentes* ou *incohérentes* (byzantines), en termes de *perception des défaillances par plusieurs utilisateurs*,
- les défaillances de *bénignes* à *catastrophiques*, en termes de niveau de *sévérité* ou de *gravité* des conséquences des défaillances sur l'environnement (cf. détails au § 1.2.1).

1.3.3 Moyens

Les **moyens** sont les méthodes et les techniques à appliquer aux entraves, généralement de manière combinée, pour obtenir (ou apprécier) le niveau requis pour les attributs pour un système donné. Ils sont classifiés en 4 types :

- ***la prévention de fautes*** pour empêcher l'introduction ou l'occurrence de fautes dans le système,
- ***l'élimination de fautes*** pour réduire leur présence dans le système

Évitement

- ***la prévision de fautes*** pour estimer leur présence et leurs conséquences sur le service offert,
- ***la tolérance aux fautes*** pour permettre au système de remplir sa mission en dépit des fautes.

Acceptation

La *prévention* et *l'élimination* sont des moyens ***d'évitement*** de fautes, car ils visent à rendre le système exempt de fautes. La *prévision* et la *tolérance* sont des moyens ***d'acceptation***, car ils visent à accepter de vivre avec un système comportant des fautes.

1.3.4 Éléments de sûreté de fonctionnement pour nos travaux

Le cadre général des définitions de la sûreté de fonctionnement ayant été fixé, précisons maintenant, par rapport à ce qui a été dit au § 1.2, sur quels aspects portent nos travaux pour les SCC, et plus précisément pour leur système de communication.

Concernant les attributs pour ces systèmes, ils sont tous importants, sauf la sécurité-confidentialité, qui reste encore à ce jour considérée comme moins prioritaire. Pour notre part, nous focaliserons sur la sécurité-innocuité, et plus particulièrement sur l'intégrité, qui est, rappelons le, un pré requis pour la sécurité-innocuité.

Nos travaux visent à proposer des moyens de se prémunir contre des défaillances catastrophiques en valeur ou temporelles, en considérant que les fautes à l'origine de ces défaillances sont des fautes opérationnelles externes ou internes, temporaires ou permanentes.

Enfin, les moyens de sûreté de fonctionnement que nous cherchons à mettre en place sont de type tolérance aux fautes au niveau d'un système de communication.

1.4 Intégrité des communications

Nous allons donc maintenant, dans une première phase, définir les propriétés à satisfaire pour garantir une communication dite intègre dans les SCC considérés. Pour cela, nous présentons d'abord l'ensemble des propriétés d'un point de vue général. Puis, sur la base de cet ensemble, et des caractéristiques des systèmes considérés au § 1.2, qui vont restreindre cet ensemble, nous présentons les propriétés que nous retiendrons dans le cadre de nos travaux. Dans une seconde phase, nous présentons les principes des mécanismes usuels pour assurer ces propriétés, en focalisant plus particulièrement sur les codes détecteurs d'erreurs, pour des raisons que nous justifierons dans la section 1.4.2.1.

1.4.1 Propriétés d'une communication intègre

Définir les propriétés d'intégrité d'une communication nécessite de donner d'abord les éléments et définitions d'un réseau de communication sur lesquels nous nous appuyons.

1.4.1.1 Définitions pour un réseau de communication

Un réseau de communication sert à échanger des données entre un nœud émetteur et un ou des nœuds récepteurs. Dans un SCC, nous considèrerons que le nœud émetteur sera généralement le calculateur élaborant les ordres de commande, et que le nœud récepteur peut être, soit un nœud rattaché à des actionneurs ou des capteurs, soit un nœud intermédiaire (ou d'interconnexion), en particulier dans le cas d'un SCC avec des nappes d'actionneurs.

Pour définir les propriétés d'une communication intègre, nous ne nous intéressons qu'au réseau de communication proprement dit. De ce fait, ces définitions ne sont soumises à aucune considération particulière sur les défaillances des nœuds extrêmes émetteur et récepteur de la chaîne de commande-contrôle. Par exemple, nous ne nous intéressons pas aux problèmes liés à la génération ou à l'exécution erronée des ordres. Les seules défaillances considérées, sources d'altération de l'intégrité, sont celles du réseau d'interconnexion faisant intervenir, en plus des canaux de communication, un ou plusieurs nœuds intermédiaires dans le cas d'un réseau de communication complexe permettant de gérer des nappes de microsystèmes (cf. figure 1.4).

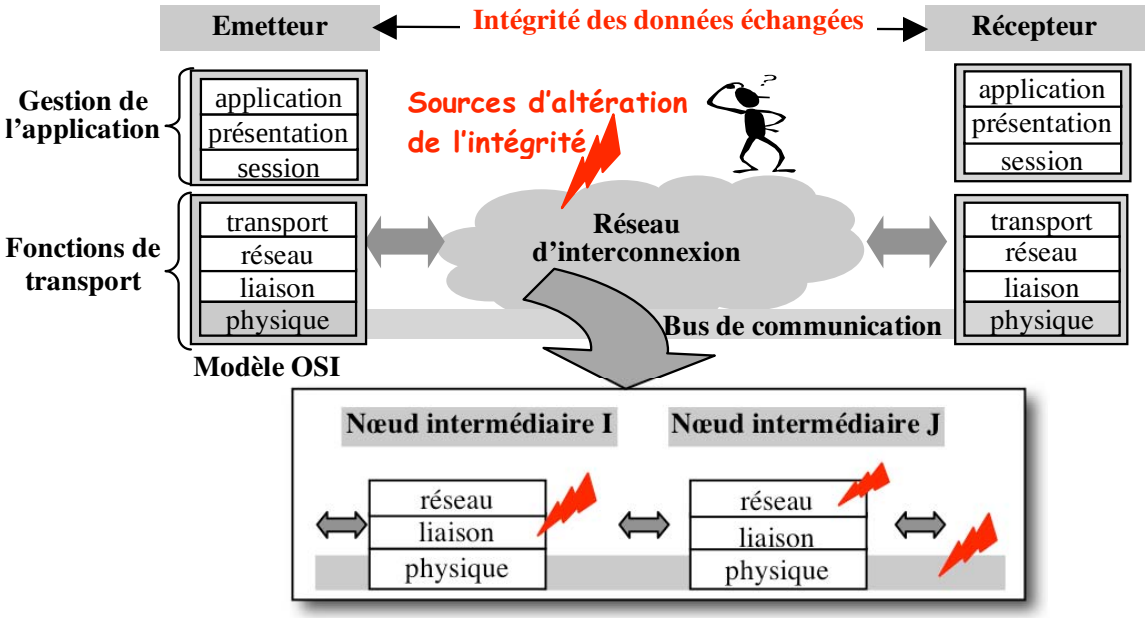


Figure 1.4 - Schéma simplifié d'un réseau de communication

L'échange de données au travers d'un réseau de communication s'appuie sur un standard de communication basé généralement sur le modèle OSI en 7 couches. Fonctionnellement, les 7 couches peuvent être divisées en deux familles bien distinctes : les couches 1 à 4 qui offrent les services de communication (transfert de données), et les couches 5 à 7 qui offrent les services d'application (traitement des données). Les réseaux de terrain ne mettent généralement en œuvre que les couches 1 à 3 relatives au transfert de données ainsi qu'à la couche application.

Dans le cas d'une liaison directe entre nœuds émetteur et récepteur, l'essentiel des informations véhiculées sur le bus de communication est la donnée correspondant à la commande. Dans le cas d'un bus, ou d'un réseau maillé avec des nœuds intermédiaires effectuant des opérations de commutation ou de routage, cette donnée est encapsulée dans un message contenant davantage de champs, dont au moins un champ adresse du nœud destinataire de la donnée.

1.4.1.2 Propriétés générales d'une communication intègre

Pour le réseau de communication par l'intermédiaire duquel des abonnés s'échangent des données, il est nécessaire de garantir, avec une probabilité donnée, que ce qui est émis soit correctement délivré au récepteur, c'est-à-dire que des propriétés d'intégrité soient satisfaites. Nous commençons par présenter ci-après ces propriétés de manière générale, sans prendre en compte les spécificités liées à un type d'application donné.

Les propriétés d'intégrité peuvent être définies en ne considérant que l'échange d'un seul message ou en considérant un ensemble de messages. Dans le premier cas, on s'intéresse à *l'intégrité d'un ou plusieurs champs* dans un même message ; dans le second cas, on parlera de *l'intégrité d'une séquence.*

Pour présenter ces propriétés, nous avons choisi de les regrouper en quatre classes, selon qu'elles concernent les aspects de fiabilité, de routage, d'ordre ou temporels. Les propriétés relatives à la fiabilité et à l'ordre sont couramment utilisées pour caractériser la qualité de service (QoS en anglais) de protocoles Internet. On parle ainsi de protocoles à fiabilité totale et à ordre total, ou à fiabilité partielle et/ou à ordre partiel.

Selon les propriétés requises au niveau d'un réseau de communication, les fonctions de protection (code détecteur, code correcteur, répétition, etc.) utilisées, ou les protocoles mis en œuvre, pourront être différents. Ces fonctions seront évoquées au § 1.4.2 et les différents types de protocoles (ex. : asynchrone, synchrone) dans le chapitre 2.

a) Propriétés rattachées à la fiabilité

Elles concernent en général l'intégrité du champ de données et peuvent s'énoncer comme suit.

P1 : pour tout message reçu, si le champ "données" est différent de ce qui a été émis, il doit pouvoir être rendu identique à ce qui a été émis

Cette propriété est très forte et nécessite l'utilisation de codes correcteurs d'erreurs et/ou la duplication de voies de transmission. Dans tous les cas, les solutions sont très coûteuses, et ne sont pertinentes que si le nombre de bits altérés est faible.

P2 : pour tout message reçu, si le champ "données" est différent de ce qui a été émis, l'altération doit être décelable

Cette propriété est moins forte que la précédente et son respect est assuré par l'utilisation de codes détecteurs d'erreurs ; le nombre de bits altérés qui peuvent être traités est plus grand que dans le cas précédent à niveau de redondance équivalent.

P3 : à tout message émis doit correspondre un message reçu

Cette propriété exprime le fait que la perte de messages n'est pas admise. La perte peut, par exemple, provenir du rejet d'un message détecté comme erroné au niveau d'un nœud intermédiaire et pour lequel il n'y aurait pas eu répétition (duplication).

P4 : tout message reçu doit correspondre à un message émis

Cette propriété exprime le fait que la répétition d'un message par le réseau d'interconnexion doit être évitée. Cette répétition pourrait, par exemple, être due à l'altération d'un message d'acquittement retourné par le nœud récepteur.

b) Propriétés rattachées au routage

Les propriétés précédentes ne concernent que des échanges ne faisant intervenir qu'un seul nœud émetteur et un seul nœud récepteur. Nous devons maintenant aborder le cas où un nœud peut émettre des messages vers plusieurs nœuds récepteurs, ainsi que le cas où un nœud peut recevoir des messages de plusieurs nœuds émetteurs.

Le premier cas conduit à la propriété suivante :

P5 : tout message émis doit être délivré (acheminé) au bon destinataire

Le second cas conduit à la propriété suivante :

P6 : un nœud récepteur ne doit recevoir de messages que des nœuds émetteurs censés communiquer avec lui, ou tout du moins, il doit être capable d'identifier l'émetteur

c) Propriété rattachée à l'ordre (intégrité d'une séquence)

Les propriétés exprimées en a) et en b) ne concernent qu'un seul message, qu'il s'agisse de l'intégrité du champ de donnée, ou celle des champs adresse. La prise en compte d'un **lot de messages** conduit à la définition de l'intégrité d'une séquence :

P7 : l'ordre de réception des messages au niveau d'un récepteur doit respecter l'ordre d'émission

Cette propriété, telle qu'exprimée, n'exclut pas la perte de messages, ni la duplication de messages sous certaines conditions. Une garantie de l'intégrité totale d'une séquence, c'est-à-dire une séquence reçue qui soit identique à la séquence émise, exige le respect des propriétés, P3, P4 et P7. Le non-respect de l'ordre peut résulter d'altérations au sein d'un réseau maillé, ou au stockage des messages dans un nœud intermédiaire et réémission dans un ordre incorrect.

d) Propriétés temporelles

Les propriétés définies en a), b) et c) permettent de garantir que les données traitées par un nœud récepteur soient correctes en valeur. Dans le cas de SCC critiques temps réel, il faut de plus que ces valeurs soient traitées en temps voulu, ce qui conduit aux propriétés suivantes.

P8 : le délai d'acheminement (la latence) d'un message entre émetteur et récepteur doit être inférieur à une valeur dictée par les contraintes temps réel de l'application

Cette propriété peut être garantie par l'utilisation de protocoles synchrones de communication, qui évitent toute forme de contention lors de l'accès à un bus.

P9 : un nœud récepteur ne doit pas consommer des données qui ne respectent pas un certain degré de "fraîcheur", c'est-à-dire avec une date de péremption dépassée

L'énoncé de cette propriété résulte du fait que, appliquer une commande, qui aurait stagné un certain temps dans le réseau d'interconnexion, peut ne plus avoir de sens selon l'évolution du système commandé, et même présenter un certain danger. Cette propriété est obligatoirement respectée si la propriété précédente est respectée ; elle exprime des conditions moins fortes.

1.4.1.3 Propriétés d'intégrité retenues dans le cas des systèmes considérés

L'ensemble de propriétés énoncées précédemment peut être très difficile à garantir lorsque l'on prend en compte les différentes sources d'altération qui peuvent affecter des échanges de messages au travers d'un réseau d'interconnexion avec des nœuds intermédiaires agissant au moins au niveau de la couche 2. Toutefois, les caractéristiques des systèmes considérés, comme leur dynamique, peuvent permettre de s'accommoder du non-respect de certaines propriétés ; le type de commande utilisé (valeur relative ou valeur absolue) peut également avoir un impact sur les propriétés à satisfaire.

Par ailleurs, les propriétés à considérer sont différentes suivant que l'on cherche à garantir, avec un certain niveau, une disponibilité de la commande et/ou sa sécurité-innocuité. Dans le cadre de nos travaux, nous nous sommes plus particulièrement intéressés à la sécurité-innocuité, c'est-à-dire à garantir une commande sûre.

Prenons maintenant en compte les caractéristiques principales des systèmes considérés : commande en valeur absolue et système à dynamique lente (cf. § 1.2.2). Le fait d'utiliser une commande en valeur absolue ne rend pas nécessaire le respect des propriétés sur la perte ou duplication de messages (propriétés P3 et P4), ni le respect de l'intégrité d'une séquence (propriété P7). Un système à dynamique lente ne nécessite pas que toute altération de la donnée contenue dans un message soit décelable (propriété P2), et donc a fortiori n'exige pas le respect de la propriété P1. Nous devons en fait garantir que la probabilité de non-détection de $\boldsymbol{X}$ messages erronés dans un lot de $\boldsymbol{N}$ messages est inférieure au seuil de criticité visé.

La mise en œuvre d'une commande sûre exige, bien entendu, le respect des propriétés rattachées au routage (P5 et P6). Le fait de considérer des SCC temps réel exige de garantir une certaine "fraîcheur" des données (P9), la propriété concernant le délai d'acheminement (P8) n'étant à considérer que pour garantir un certain niveau de disponibilité.

Par la suite, au lieu de se référer aux propriétés requises, nous ferons référence aux types d'erreurs qu'elles exigent de traiter. Ainsi, pour garantir la propriété P1, il faut détecter les *erreurs en valeur*. Pour le respect de P5 et P6, il faut assurer la détection des *erreurs d'adressage*, tandis que le respect de P9 nécessite de détecter les *erreurs temporelles*.

1.4.2 Mécanismes pour assurer l'intégrité des communications

Dans cette section, nous ne cherchons pas à couvrir tous les mécanismes nécessaires pour garantir l'ensemble des propriétés d'intégrité que nous avons retenues au §1.4.1.3. Nous nous focalisons sur les propriétés d'intégrité des données, et plus précisément sur la détection des altérations des données (propriété P2), qui est la propriété essentielle à garantir dans le cadre de nos travaux. Nous traiterons donc plus particulièrement de la détection des erreurs en valeur.

La méthode la plus usitée pour assurer une intégrité des données est d'utiliser des codes de protection contre les altérations : des codes uniquement détecteurs d'erreurs ou également correcteurs. Les codes utilisés diffèrent relativement selon que l'on s'intéresse à la protection de blocs de données dans un espace mémoire ou à la protection de données transmises sur un canal de communication. En effet, les modèles d'erreurs peuvent être différents : erreur de ligne ou de colonne pour une mémoire, erreur de paquet dans une transmission. La façon de mettre en œuvre les codes peut aussi différer, puisque pour la protection d'un canal de transmission, les opérations de codage et de contrôle à la volée sont privilégiées.

Pour les codes détecteurs et correcteurs, la théorie fondamentale du codage algébrique de l'information introduite par Peterson [Peterson 1961, Peterson & Weldon 1972] constitue une base importante pour la protection dans les communications numériques.

Dans la suite de cette section, après une brève introduction générale aux codes de protection contre les erreurs, nous focalisons sur les codes détecteurs d'erreurs et plus particulièrement sur les codes CRC. En effet, ce type de code est largement utilisé par les équipements réseau d'un système de communication numérique, et de plus, la première partie de nos travaux a consisté à analyser si les performances de capacité de détection de tels codes permettaient de satisfaire les objectifs de sûreté de fonctionnement spécifiés pour notre cas d'étude (cf. § 1.5.4.3).

1.4.2.1 Généralités sur les techniques de protection contre les erreurs

Les protections contre les erreurs de transmission dans un réseau de communication se basent sur des techniques de détection et de correction (cf. figure 1.5).

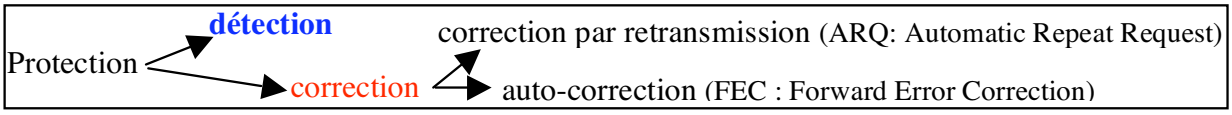


Figure 1.5 - Techniques de protection contre les erreurs

a) Détection

Pour se protéger, il faut au moins pouvoir détecter. Pour cela, un émetteur, qui veut transmettre un message (suite binaire quelconque), transforme d'abord le message initial à l'aide d'un procédé donné de calcul (ou de codage) qui génère et introduit une certaine redondance des informations dans le message codé résultant. Grâce à cette redondance, le récepteur vérifie avec le même procédé de calcul (ou de décodage) que le message reçu est bien celui envoyé.

Après détection d'une erreur, le message peut être transmis à la couche application avec une indication d'erreur. Mais il peut aussi ne pas arriver au nœud récepteur si la détection a lieu dans un nœud intermédiaire ; cela se traduit par la perte de message pour la couche application.

Des stratégies de recouvrement peuvent alors être mises en place au niveau de l'applicatif : par exemple, utiliser le dernier message reçu dans le cas d'un système de commande-contrôle. La stratégie utilisée dans le cadre de notre cas d'étude est présentée dans le chapitre 2.

b) Correction

La correction des erreurs de transmission peut s'effectuer par auto-correction à la seule charge du récepteur, ou par retransmission impliquant à la fois le récepteur et l'émetteur.

Dans le cas de l'**auto-correction**, la redondance d'information doit être suffisante pour pouvoir reconstituer le message initial à partir du message reçu. Cela est possible par l'utilisation de codes évolués parmi lesquels nous pouvons citer les codes de Hamming, les codes BCH (*Bose-Chaudhuri-Hockenghem*) dont les codes *Reed-Solomon,* et les turbo-codes. Ces codes sont plus coûteux que les codes seulement détecteurs.

Dans le cas d'une **correction par retransmission**, l'utilisation de codes détecteurs suffit. Lorsque le récepteur détecte une erreur, il demande à l'émetteur de retransmettre une nouvelle fois le message. Ce principe est généralement préféré dans les réseaux où le taux d'erreur est faible et le délai de retransmission est tolérable, c'est-à-dire quand il est compatible avec les contraintes temps réel de l'application. Il est utilisé dans de nombreux protocoles (ex. : HDLC, X25, TCP/IP). La correction par retransmission est une stratégie de recouvrement à la détection d'une erreur, mais elle est ici mise en œuvre au niveau des couches de transfert de données.

D'une manière générale, les procédés de correction nécessitent un niveau de redondance nettement plus élevé que celui nécessaire pour de la simple détection. Dans le cadre de nos travaux, nous écarterons donc les codes correcteurs, même si certains sont très populaires et souvent utilisés en pratique, comme les turbo-codes ou les codes Reed-Solomon. Mais, en plus d'un surcoût élevé par rapport aux codes détecteurs, les codes correcteurs ne sont pas adaptés au type de protection que nous recherchons. D'une part, dans le type d'applications que nous visons, on ne cherche pas à détecter unitairement chaque message erroné, et il est donc inutile d'avoir à subir le coût des codes correcteurs. D'autre part, les erreurs vis-à-vis desquelles on cherche à se protéger peuvent avoir, avec une forte probabilité, une multiplicité (nombre de bits affectés) telle, qu'il est impossible d'effectuer une correction.

Ainsi, par la suite, pour assurer l'intégrité des communications, nous ne considèrerons que les mécanismes basés sur des codes détecteurs d'erreurs associés à des stratégies de recouvrement au niveau applicatif.

1.4.2.2 Définitions générales pour les codes détecteurs d'erreurs

Le principe de base pour détecter une erreur de transmission est d'ajouter des bits de contrôle aux bits qui constituent le champ de données. L'idée est donc de transmettre $n = k+p$ bits de données au lieu des k bits de données utiles, et d'utiliser les p bits supplémentaires pour associer à l'émission un code de contrôle aux données utiles, et vérifier la validité de ce code à la réception pour détecter une erreur éventuelle.

On appelle **mot du code**, la suite des n bits obtenue après un codage (k, n) : k est la longueur initiale des mots, et le nombre n de bits composant un mot du code est la **longueur du code**.

Un code (k, n) est dit **systématique** s'il existe k bits du mot de code qui sont égaux aux k bits de données utiles. Les p ($p = n-k$) autres bits forment un **champ de contrôle d'erreur**. Le rendement du code est : $R = k/n$ et il caractérise le débit utile des données transmises par rapport à leur débit réel.

Une notion importante, qui caractérise un code et qui a un fort impact sur sa capacité de détection, est la notion de **distance de Hamming d'un code** elle-même définie à partir de la notion de **poids de Hamming d'un mot**, qui est le nombre de bits à 1 qu'il contient.

La *distance de Hamming d'un code* est la distance minimale entre tous les mots du code, la distance entre deux mots de même longueur étant définie par le nombre de positions binaires qui diffèrent de valeur entre ces deux mots [Kazakov 2001]. A titre d'illustration :

- le poids de Hamming de 01001100 est égal à 3,
- la distance de Hamming entre 01001100 et 010**10**10**1** est de 3.

La **capacité de détection** d'un code est définie par les configurations erronées qu'il est capable de détecter. Pour ces configurations, on définit une erreur simple (respectivement double, ou d'ordre *x*) comme une erreur affectant une seule (respectivement 2, ou *x*) position(s) binaire(s) d'un mot. Pour qu'un code ait une capacité de détection des erreurs d'ordre *e*, sa distance de Hamming doit être supérieure ou égale à *e+1* [Fujwara *et al.* 1985]. Par exemple, un code avec une distance de Hamming de 3, a une capacité de détection d'erreurs inférieure ou égale à 2.

1.4.2.3 Les codes CRC

Nous présentons maintenant les éléments de base sur les codes CRC auxquels nous ferons référence dans la suite de ce mémoire. En effet, ce sont ces codes que nous avons étudiés, car ce sont les plus usuellement employés dans les réseaux de terrain. Après avoir situé les codes CRC parmi les codes détecteurs, nous en présentons le principe, puis le pouvoir de détection associé et nous terminons cette section en traitant de leur mise en œuvre.

1.4.2.3.1 Positionnement des codes CRC

Il existe de nombreux codes pour protéger un canal de transmission. La figure 1.6 en donne une classification et permet de situer les codes CRC qui sont une sous-classe des codes cycliques.

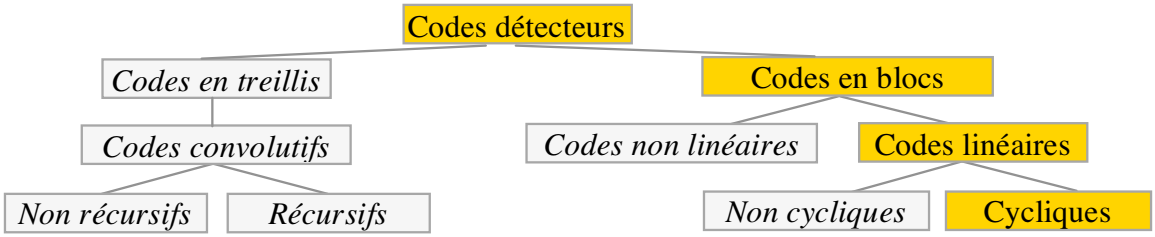


Figure 1.6 - Codes CRC et codes détecteurs d'erreurs

Dans les codes en blocs, le codage et le décodage ne dépendent que des informations dans un bloc à transmettre, et non, comme dans le cas des codes en treillis, d'informations dans les blocs précédemment transmis. Un code en bloc de taille *T* et de longueur *n*, défini sur un alphabet de *q* symboles (1 et 0 pour le codage binaire par exemple), est un ensemble de *T* séquences (vecteurs) *q*-aires de longueur *n* appelées mots de code. Généralement, $T=q^k$, *k* étant un entier. Le code est alors désigné par la paire (*n, k*), où *k* représente la dimension du code.

Les ***codes linéaires*** sont des codes dont chaque mot du code (noté **C**) est obtenu après transformation linéaire des bits du mot initial (noté ***I***). Ces codes sont caractérisés par leur matrice génératrice ***G(k, n)*** telle que : ***I.G = C***. Par exemple :

$$\mathbf{I}\,(1*3)\ .\ \mathbf{G}\,(3*4) = \mathbf{C}\,(1*4) \quad \Leftrightarrow \quad [\,1\ 0\ 1\,]\ .\ \begin{pmatrix} 1\ 0\ 1\ 1 \\ 0\ 1\ 0\ 1 \\ 1\ 1\ 0\ 1 \end{pmatrix} = [\,0\ 1\ 1\ 0\,]$$

Le récepteur utilise une autre matrice ***H(n-k, n)***, appelée matrice de contrôle, pour savoir si un mot reçu **C'** est un mot du code. La matrice ***H(n-k, n)*** se construit en se basant sur l'équation suivante : $\boldsymbol{G}(k, n)\ .\ \boldsymbol{H}^T(n, n-k) = \boldsymbol{0}\ (k, n-k)$, où H^T est la transposée de la matrice *H*.

La détection d'erreurs se base sur le calcul du syndrome ***S*** du mot reçu, et qui représente le produit matriciel : $\boldsymbol{C'.H^T}$. Si le syndrome du mot est nul, alors ce mot appartient au code et le récepteur considère donc qu'il n'y a pas eu d'erreur de transmission [Funk 1996].

Un ***code cyclique*** (*k, n*) est un code linéaire (*k, n*) tel que toute permutation circulaire d'un mot du code est encore un mot du code. Par exemple, un code cyclique (1, 2) possède les mots de code suivants : {01, 10} ou {00, 11} mais pas {01, 11}. Et un code cyclique (1, 3) possède les mots de code suivants : {000, 111}. Cette classe de codes détecteurs d'erreurs contient une sous-classe de codes très populaires, les *Codes à Redondance Cyclique* (*CRC*).

1.4.2.3.2 Principe des codes CRC - utilisation pour la détection d'erreurs

Comme pour tous les codes détecteurs dits systématiques, le principe de base utilisé dans les codes CRC pour détecter une erreur est d'associer des bits de contrôle aux bits de données. L'idée est donc de transmettre *k+p* bits de données au lieu des *k* bits de données utiles, les *p* bits supplémentaires permettant d'associer à l'émission aux données utiles, un code de contrôle appelé champ CRC, dont le récepteur vérifie la validité pour détecter une erreur éventuelle. Le champ CRC est construit à l'émission à partir d'un polynôme générateur *G(x),* que tous les récepteurs utilisent pour effectuer leur vérification. La procédure est la suivante :

- le flot des *k* bits de données utiles est interprété comme un polynôme *M(x*) de degré *k-1* avec des coefficients 0 et 1 selon que le bit correspondant est respectivement à 0 ou à 1 ;
- le polynôme *M(x)* est alors multiplié par x^p, où *p* est le nombre des bits de contrôle ;
- le polynôme $M(x).x^p$ est alors divisé (modulo 2) par le polynôme générateur *G(x),* et le reste de cette division est un polynôme noté *R(x),* qui fournit alors les bits de CRC ;
- le nœud émetteur transmet les bits de données correspondants à la représentation polynomiale de $x^p.M(x)+R(x)$ et ainsi, **ce qui est émis est donc un multiple de *G(x)*** ;
- le polynôme reçu par le nœud destinataire est divisé modulo 2 par *G(x)* : un reste nul traduit l'absence d'une erreur, alors qu'un reste non nul indique la présence d'une erreur, puisque par construction, le polynôme émis est divisible par *G(x).*

1.4.2.3.3 Capacité de détection des codes CRC

La capacité de détection des codes CRC dépend du polynôme générateur *G(x)* commun à tous les émetteurs et récepteurs. Mais il faut se rappeler que, par construction même, les erreurs égales à des polynômes divisibles par *G(x)* ne sont pas détectées [Baicheva *et al.* 1988].

Les résultats de base sont qu'un code CRC, engendré par un polynôme générateur de degré *q*, de distance de Hamming *d*, détecte comme configurations d'erreur :

- tout bloc de bits erronés de taille strictement inférieure à la distance de Hamming du code (soit *d-1*) correspondant à des erreurs en rafale ou *burst* ;
- toute erreur correspondant à un nombre impair de bits erronés (sous réserve de certaines propriétés du polynôme générateur qui seront décrites par la suite) ;
- toute erreur multiple dont la multiplicité (nombre de bits erronés) est strictement inférieure au degré *q* du polynôme générateur G(x).

L'occurrence de ces configurations d'erreur dépend de la qualité du canal de transmission caractérisée par le taux moyen de bits erronés (BER : Bit Error Rate). Globalement, la capacité de détection d'un code CRC est déterminée par la probabilité ***Pue*** de non-détection d'erreurs du canal. Nous y reviendrons au chapitre 2.

1.4.2.3.4 Mise en œuvre matérielle des codes CRC

Les codes CRC sont généralement mis en œuvre de manière matérielle sous forme d'unités CRC dans les différents équipements d'un système de communication tels que les contrôleurs de communication. À titre d'exemple, la figure 1.7 décrit la position des unités CRC au sein d'un contrôleur de communication pour le protocole TTP/C.

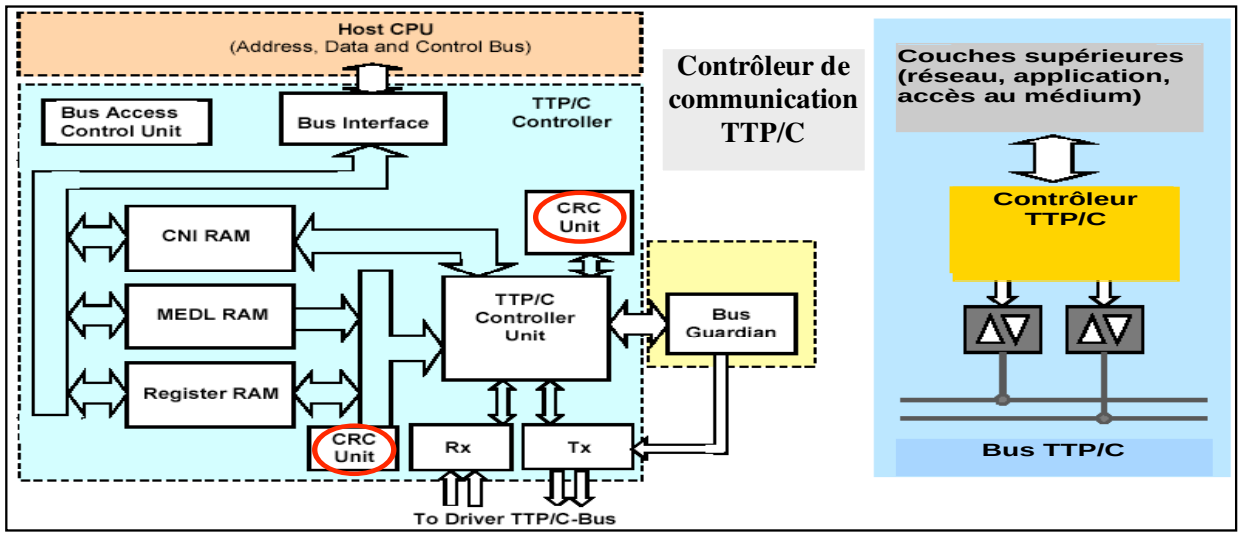


Figure 1.7 - Contrôleur de communication TTP/C

La mise en œuvre des unités CRC est simple au niveau de la conception et de la réalisation, vu qu'elle se base sur des registres à décalage et des circuits logiques "OU exclusif", dont le nombre et la disposition dépendent du polynôme générateur utilisé. La figure 1.8 décrit une mise en œuvre dans le cas d'un code CRC sur 8 bits, pour le polynôme générateur $x^8+x^5+x^4+1$.

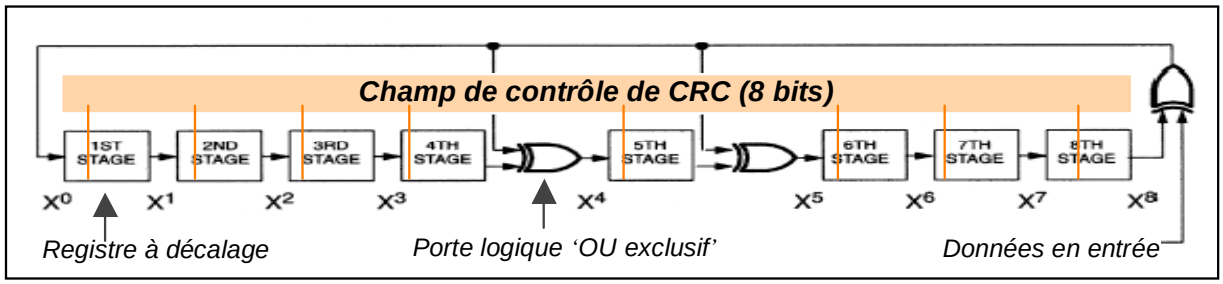


Figure 1.8 - Exemple de mise en œuvre de l'unité CRC

Cette simplicité de mise en œuvre, qui de plus n'introduit pas de délai supplémentaire à la latence de transmission des données [Henriksson & Liu 2003], et enfin les bonnes capacités de détection des différents types d'erreurs qu'offrent les codes CRC, les ont rendus très populaires auprès des constructeurs. À titre d'exemple, nous citons leur utilisation dans les réseaux de communication usuels (Ethernet, CAN, TTP, etc.), ainsi que leur adoption dans différents standards (ATM, IEC, IEEE, CCITT, IBM-SDLC) [Boudreau *et al*. 1994].

Mais il existe aussi des réalisations logicielles des codes CRC, qui offrent plus de flexibilité, notamment au niveau du choix des polynômes générateurs. Nous analyserons cette possibilité dans le cadre de la proposition des compléments de codage que nous faisons dans le chapitre 3, et dont une mise en œuvre particulière à base de codes CRC est analysée dans le chapitre 4.

1.5 Cas d'étude : les systèmes de CDV

La thématique de nos travaux est issue d'une application du domaine de l'aviation civile : les *systèmes de Commandes De Vol (CDV)*, dont le rôle est de contrôler la trajectoire de l'avion en agissant sur les gouvernes, lors du décollage, du vol et de l'atterrissage.

Ce type de système est particulièrement illustratif à plusieurs titres, puisqu'il est :

- réel et étudié en collaboration avec un industriel dans le cadre d'un contrat de recherche,
- largement diffusé, car présent dans tous les avions,
- représentatif sur bien des points de SCC dans d'autres domaines que celui de l'aviation.

Après avoir présenté la place des CDV dans l'avionique et les fondements de leur fonctionnement (forces utilisées et gouvernes pour les contrôler), nous décrivons les technologies actuelles assurant ce fonctionnement, puis celles envisagées pour le futur. Nous précisons alors comment se décline notre problématique sur ce type de système.

Les définitions qui suivent se limitent à ce qui est nécessaire pour nos travaux, sur la base des avions civils usuels, sans particularités telles qu'une motorisation pour une poussée verticale ou une disposition particulière de surfaces (ex. : un stabilisateur horizontal placé devant les ailes). Enfin, certains points sont des solutions adoptées sur les avions de type Airbus.

1.5.1 Place et rôle des CDV dans le monde de l'avionique

Les CDV sont un des systèmes de ***l'avionique***, sachant que ce terme couvre l'ensemble des logiciels et des matériels électriques et électroniques (calculateurs, bus, capteurs, actionneurs) à bord d'un avion et impliqués dans son fonctionnement. Le spectre des fonctionnalités est donc très large. Citons le contrôle des gouvernes, des moteurs, du train d'atterrissage, de la pressurisation, ainsi que la gestion des énergies (électrique, hydraulique) et du carburant. Mais également, en relation avec l'extérieur, les communications sol-bord, etc.

Les CDV sont en charge du contrôle des *gouvernes* (ou *surfaces de contrôle*) pour contrôler la trajectoire de l'avion à partir des variations d'intensité et de distribution des forces appliquées à l'avion. Aujourd'hui, c'est à partir de la consigne du pilote et d'informations issues de capteurs que plusieurs calculateurs déterminent en permanence les ordres, dit de *braquage*, à appliquer aux actionneurs des gouvernes pour que celles-ci prennent la position adéquate.

Les CDV sont donc non seulement des SCC *temps réel* et *critique*s, mais également *"**embarqués et mobiles**"*. Cette dernière caractéristique traduit un contexte clos, et entraîne donc des contraintes supplémentaires, principalement lors de leur fonctionnement. Il s'agit classiquement de contraintes d'autonomie de traitement et d'énergie, de poids, de dimensions, etc., et donc aussi de sûreté de fonctionnement, car de tels systèmes doivent être particulièrement robustes du fait de leur inaccessibilité. Quant au caractère *critique*, il provient du fait que certaines défaillances peuvent avoir des effets catastrophiques sur le plan humain, voire économique : typiquement, certaines pertes de contrôle de l'avion ou des efforts exceptionnels sur la structure de l'avion peuvent conduire à la destruction de l'avion. Pour toutes ces raisons, ces systèmes sont en plus soumis à de sévères *règlements de certification*.

1.5.2 Fondements des CDV : forces et gouvernes

1.5.2.1 Forces appliquées à un avion

Un avion est principalement soumis à quatre forces repérées par rapport aux trois axes de l'avion : *axe de roulis* (axe longitudinal), *axe de profondeur ou de tangage* (axe transversal), et *axe de lacet* (perpendiculaire au plan formé par les deux premiers axes) (cf. figure 1.9). Sans détailler leurs corrélations, ces forces sont :

- le *poids* : force liée à la masse de l'avion et qui l'entraîne vers le sol selon l'axe de lacet,
- la *portance* : force opposée au poids et qui résulte de la pression de l'air sur toute la surface de l'avion, les ailes jouant donc un rôle prédominant,
- la *poussée* : force produite par les moteurs et qui agit principalement sur le déplacement horizontal de l'avion selon l'axe de roulis,
- la *traînée* : force opposée à la *poussée* et qui résulte de la résistance de l'air au mouvement de l'avion, en fonction essentiellement de la forme de la surface de l'avion, de sa vitesse et de la viscosité de l'air.

Au niveau du système de CDV, n'interviennent dans la gestion de ces forces que les surfaces de l'avion, que nous allons donc détailler maintenant.

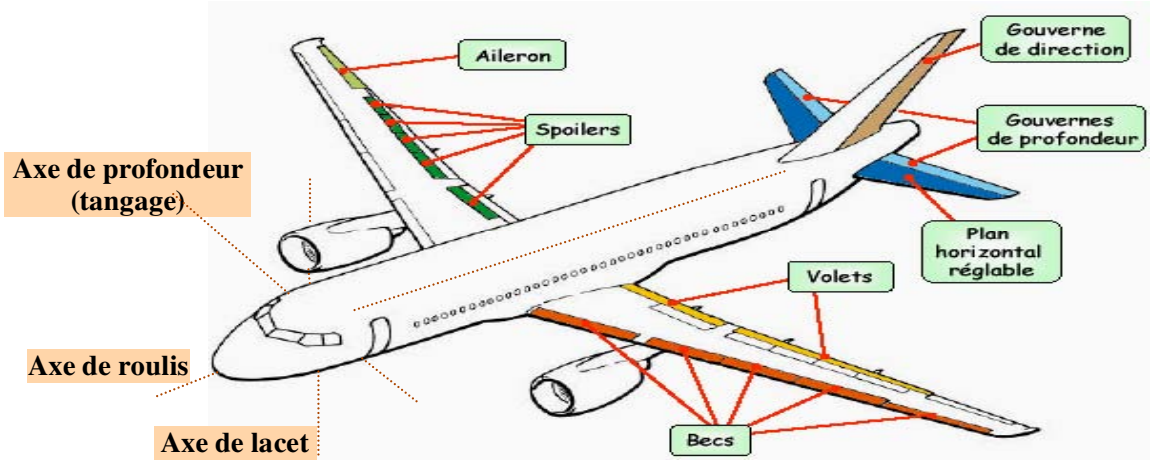


Figure 1.9 - Les axes de forces et les surfaces de contrôle

1.5.2.2 Surfaces de contrôle

Toutes les surfaces n'ont pas le même niveau d'influence pour le contrôle de la trajectoire de l'avion : certaines sont donc plus *critiques* que d'autres. Mais surtout, ce contrôle ne peut se faire que parce que certaines surfaces sont *mobiles :* c'est en "jouant" sur leur position que la trajectoire peut être modifiée. Ces sections mobiles sont donc aussi appelées *surfaces de contrôle*. Les surfaces les plus impliquées sont les ailes, contribuant le plus à la portance, et la queue, servant à maintenir l'avion droit en vol. La figure 1.9 montre les principales surfaces.

1.5.2.2.1 Surfaces fixes et mobiles

La queue est classiquement composée, d'une part, d'un *stabilisateur horizontal* fixe, avec pour section mobile le *gouvernail de profondeur* pour contrôler la rotation autour de l'axe de tangage (inclinaison verticale de l'avion), et d'autre part, d'un *stabilisateur vertical* fixe, avec pour section mobile le *gouvernail de direction,* pour contrôler la rotation autour de l'axe de lacet (rotation horizontale vers la gauche ou la droite).

Les ailes sont équipées de plusieurs types de sections mobiles, ayant parfois des rôles doubles et combinés. Sur le bord arrière des ailes, il y a les *volets* et les *ailerons*, qui sont inclinables vers le bas et modifient l'écoulement de l'air au-dessous des ailes. Les ailerons servent surtout à faire tourner l'avion sur son axe de roulis (inclinaison latérale), alors que les volets servent surtout à modifier la portance et la traînée. Sur le bord avant, il y a les *becs*, pour modifier la portance. Enfin, sur le dessus des ailes, on trouve les ***spoilers***, qui sont inclinables vers le haut et modifient l'écoulement de l'air au-dessus de l'aile, servant d'aérofreins. À noter que les spoilers jouent parfois un rôle semblable à celui des ailerons, et réciproquement : un déploiement symétrique des surfaces de droite et de gauche freine l'avion, alors qu'un déploiement dissymétrique fait tourner l'avion.

Par la suite, l'exemple retenu de surfaces mobiles sera celui des spoilers. Intéressons nous maintenant à la criticité de ces surfaces, reliée aux notions de perte et d'embarquement de surface.

1.5.2.2.2 Perte et embarquement d'une surface mobile : surface critique ou non

La *perte* d'une surface survient lorsqu'elle n'est plus active. C'est le cas par exemple si les 2 servocommandes d'un aileron ont leurs circuits hydrauliques dépressurisés ou si les calculateurs (qui les commandent) les ont désactivées après avoir détecté des problèmes (typiquement : la surface ne répond plus correctement aux ordres). Par principe, cet aileron perdu prend une position dans le "lit du vent". Une surface perdue réduit les possibilités de contrôler l'avion, dans une proportion variable selon la surface, pouvant aller jusqu'à l'événement redouté. Dans ce dernier cas, la perte de la surface n'est pas tolérée.

L'***embarquement*** d'une surface survient pour une surface active qui s'écarte excessivement de la position souhaitée (l'embarquement n'est pas considéré comme un cas de perte de surface). L'écart est jugé en termes, à la fois, de valeur (de quelques degrés à une position en butée) et de durée (de quelques millisecondes à permanent). Selon le type de surface, et en considérant les deux cas extrêmes, on aboutit au fait que :

- un embarquement permanent et allant jusqu'à la butée, est tolérable s'il ne présente pas de risque pour la structure de l'avion et si le contrôle de l'avion peut être conservé,
- un embarquement transitoire de quelques centaines de ms, et de seulement une dizaine de degrés, n'est pas tolérable s'il présente des risques pour la structure de l'avion, même si le contrôle de l'avion n'est pas affecté du fait de la brièveté de l'embarquement.

Dans tous les cas, d'un point de vue criticité, on distingue deux types de surfaces selon que leur perte ou embarquement peut provoquer ou non un événement catastrophique.

Une surface est *non critique* si sa perte ou son embarquement est sans conséquence catastrophique (et est donc toléré) : typiquement, les spoilers et les ailerons. Par exemple, l'effet d'un spoiler "défaillant" sur une dizaine de spoilers en mode nominal est compensable par une nouvelle loi de commande calculée sur les spoilers restants.

Une surface est *critique* si sa perte ou son embarquement a une conséquence catastrophique (et n'est donc pas tolérée). C'est typiquement le cas des surfaces qui sont seules à assurer un rôle essentiel, tels que par exemple, le gouvernail de direction. Ce type de surface exige une robustesse accrue, obtenue par davantage de redondance des circuits de commande, d'activation, et de retour d'information de la surface vers le calculateur. Pour certaines surfaces, un des circuits de retour est même encore mécanique (utilisant des bielles).

1.5.3 Commandes de vol : état actuel et futur

Ayant défini les différentes surfaces de contrôle, nous pouvons maintenant décrire les moyens de les contrôler, c'est-à-dire, comment la consigne du pilote est traitée et transmise aux surfaces (moyens de commande), puis comment ces surfaces sont activées et par quel type d'énergie (moyens d'activation et puissance d'activation).

Il s'agit dans cette partie, pour bien situer les enjeux, de présenter dans le détail l'état actuel de ces moyens, et les orientations technologiques pour le futur, en particulier les communications numériques et l'introduction massive de microsystèmes, qui constituent en partie le cadre de nos travaux. Étant d'abord rappelé très sommairement les étapes technologiques antérieures :

- ***"Le tout mécanique"*** : à l'origine les consignes du pilote étaient répercutés directement aux surfaces, par des moyens totalement mécaniques (câbles, biellettes et poulies).
- ***L'activation hydraulique*** : la taille des avions augmentant, les forces exercées sur l'avion se sont accrues, nécessitant des moyens de contrôle de plus en plus puissants. Pour pallier en partie les problèmes induits de poids de ces moyens et d'efforts du pilote, il a été introduit, au niveau de l'activation des surfaces, des mécanismes de puissance hydraulique (comme la direction assistée d'une voiture).

1.5.3.1 Commandes actuelles : par signaux électriques analogiques (CDVE)

La dernière grande évolution significative, qui a conduit à l'état actuel, a été l'utilisation de commandes électriques générées par des calculateurs. Appelée ***fly by wire*** (Commandes De Vol Électriques, ou CDVE), cette technique supprime toutes les liaisons mécaniques entre le manche à balai du pilote (qui génère les consignes) et les actionneurs des surfaces de contrôle. Un tel système a été conçu par l'Aérospatiale et mis en place pour la première fois sur un Concorde. L'introduction de technologies numériques (au niveau de calculateurs) dans ce type de système date du début des années 80, mais n'a été certifiée par rapport aux contraintes de l'aviation civile qu'en 1988 avec l'Airbus A320 [Traverse *et al.* 2004]. Depuis, cette technique a été largement étendue à d'autres systèmes que les CDV, et même dans d'autres domaines que l'aviation pour s'appeler ***X by wire***. Ainsi, le domaine de l'automobile est un autre domaine où l'introduction des systèmes "X-by-Wire" est d'actualité. L'introduction de l'accélérateur électronique (Throttle-by-Wire) date de 1980, alors que l'introduction du volant électronique (Steer-by-Wire) ou du freinage électronique (Brake-by-Wire) est encore tributaire de la maturité de la technologie des alimentations nécessaires pour certains types de véhicules [Wilwert 2003].

1.5.3.1.1 Principes actuels de fonctionnement

Le déplacement du manche est relevé par des capteurs électriques, puis envoyé à un calculateur, qui, à l'aide de ***lois de commande***, établit la nouvelle position à atteindre par les surfaces suite à la demande du pilote. À partir de cette position et des informations renvoyées par les capteurs associés aux surfaces, le calculateur établit, à l'aide de ***lois d'asservissement***, ***les ordres de braquage*** à transmettre aux servocommandes des vérins hydrauliques de chaque surface (cf. figure 1.10). Périodiquement, le calculateur transmet une valeur réajustée de ces ordres, jusqu'à ce que la position demandée soit atteinte. Ces ordres sont transmis sous forme de variations de courant par liaison directe entre calculateur et actionneur. Les communications sont donc centralisées et analogiques.

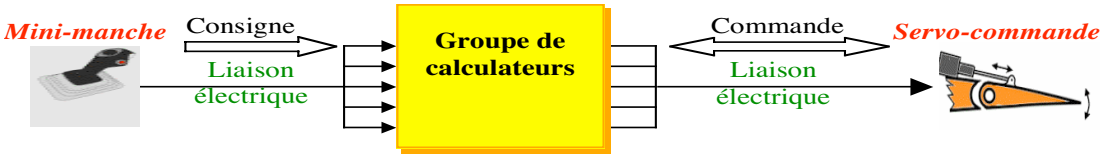


Figure 1.10 - Principe d'une commande électrique d'une servo-commande

Cette technique a permis de gagner en poids et en simplicité (suppression des liaisons mécaniques), et de réduire les efforts et la charge de travail du pilote dont les consignes sont désormais d'abord traitées par un calculateur avant d'être répercutées à la surface. Elle a aussi induit un gain notable en confort et sécurité, puisque grâce au calculateur, des *lois de pilotage* plus sophistiquées ont pu être utilisées, permettant un pilotage plus aisé, précis et sûr. En effet, si le calculateur considère qu'une consigne sort des limites du domaine nominal de vol de l'avion, il peut la moduler, voire refuser de la répercuter à la surface de contrôle [Spitzer 2000].

1.5.3.1.2 Éléments de sûreté de fonctionnement des CDVE

La sophistication des CDVE a aussi inévitablement introduit de nouveaux risques (notamment liés aux calculateurs), auxquels il a fallu apporter des réponses en termes de sûreté de fonctionnement. Nous n'en donnons que quelques grandes lignes sous l'aspect tolérance aux fautes, sur des avions de type Airbus : *groupe de calculateurs autotestables, diversification, ségrégation* [Brière & Traverse 1993, Traverse 2004].

Les calculateurs ont été conçus pour être *autotestables* vis-à-vis des fautes physiques et de conception, sur la base de détection et de compensation d'erreur sur les ordres de sortie du calculateur. Un calculateur comporte deux "voies" fonctionnellement équivalentes mais exécutant des programmes de conception différente : une voie de surveillance et une de commande (la seule à piloter les sorties physiques). Cette duplication *diversifiée* des voies permet à chaque voie de produire, à partir des mêmes entrées, un résultat indépendant. Les deux résultats sont comparés avant transmission par la voie de commande. En cas de divergence, ces sorties sont inhibées (cf. figure 1.11). Un tel calculateur est dit *fail-safe*.

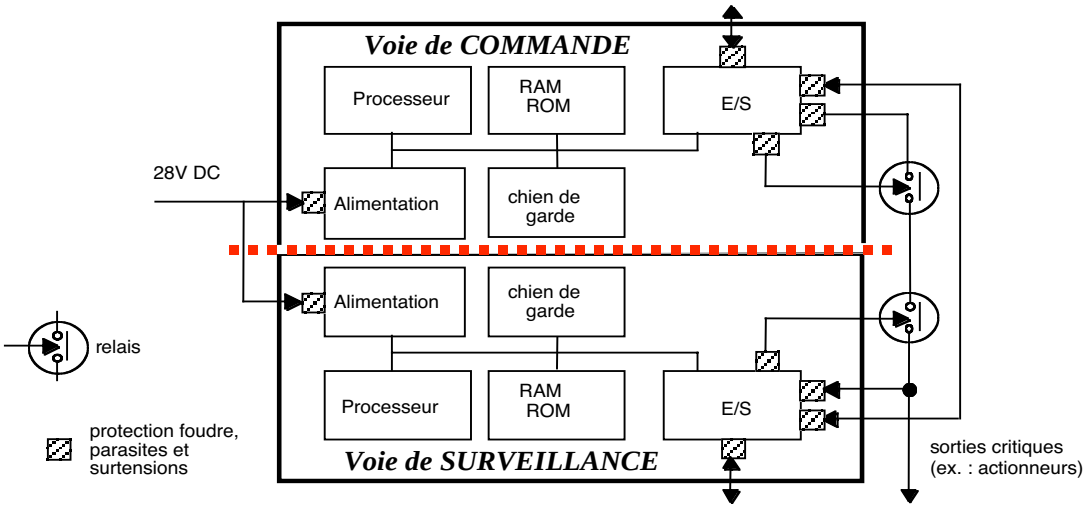


Figure 1.11 - Calculateur autotestable

Comme déjà dit implicitement, c'est un *groupe* de calculateurs, et non pas un calculateur seul qui est utilisé. Un exemple de principe d'utilisation est qu'à tout instant, tous les calculateurs sont actifs, mais un seul (appelé calculateur primaire) calcule la loi de commande.

Lorsqu'il défaille, nécessairement en mode sûr vu qu'il est autotestable, un autre calculateur du groupe, selon un ordre fixé, assure un relais immédiat en devenant le nouveau primaire.

L'utilisation d'un groupe permet de mettre en place de la *diversification* : les calculateurs sont généralement basés sur des processeurs différents et réalisés par des constructeurs différents.

Le fait que tous les calculateurs soient actifs est exploité pour réaliser de la *ségrégation* de la commande des surfaces : chacun de ces calculateurs ne contrôle qu'un sous-ensemble de surfaces, de sorte que si l'un défaille, toutes les surfaces ne soient pas affectées. À quoi s'ajoute encore une autre forme de *diversification*, liée au fait que certaines surfaces peuvent être commandées par plusieurs calculateurs (mais toujours un seul à un instant donné).

Cela donne par exemple pour les quatre calculateurs du sous-groupe commandant l'axe de tangage (cf. figure 1.12) : deux calculateurs (ELACs : *ELevator and Aileron Calculator*s) associés aux gouvernes de profondeur et aux ailerons, et deux calculateurs (SECs : *Spoiler and Elevator Calculators*) associés aux spoilers et aux gouvernes de profondeur.

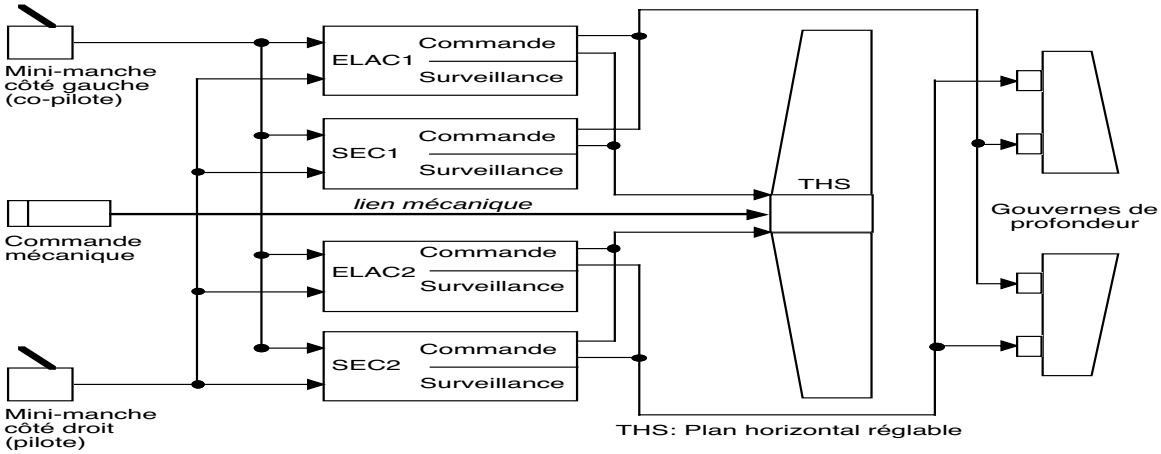


Figure 1.12 - Groupe de calculateurs pour le contrôle du tangage de l'Airbus 320

Il existe une importante redondance fonctionnelle entre les différentes surfaces de contrôle de l'avion. Cela permet de faire face à la perte totale des calculateurs associés à certaines surfaces, en admettant qu'ils défaillent de manière contrôlée. De plus, si tous les calculateurs venaient à défaillir, il reste encore un secours manuel mécanique pour commander l'avion, mais de façon plus limitée. Citons enfin une redondance d'actionneurs, qui dépend de la criticité de la surface : un seul pour les spoilers, mais deux pour les ailerons (un est actif, l'autre en stand by).

1.5.3.1.3 CDVE et Avionique Modulaire Intégrée (IMA)

Avant de conclure sur les solutions actuelles, il est impossible de ne pas évoquer le concept *d'Avionique Modulaire Intégrée* (ou *IMA* pour *Integrated Modular Avionic*s) qui, en termes de culture de conception d'un avion, introduit une forte rupture avec le passé. C'est une évolution à la charnière du présent et du futur, puisque déjà en cours d'application, notamment sur l'Airbus A380. En effet, l'avionique classique a jusqu'ici privilégié une approche spécifique, fonction par fonction, conduisant à des calculateurs complètement dédiés.

Pour réduire les coûts de développement, d'acquisition et de maintenance des systèmes à bord des avions, Airbus s'oriente vers une *banalisation des ressources de calcul* que permet le concept IMA.

Les différentes fonctions avioniques utilisent alors, et partagent même dans certains cas, des ressources de calcul génériques interconnectées par un réseau Ethernet commuté et redondé, l'AFDX (*Avionics Full DupleX ethernet*). Au sein d'une architecture IMA, chaque ressource de calcul met en œuvre [Bérard 2003] :

- un *partitionnement robuste* des ressources pour héberger des applications de niveaux de criticité différents, qui est basé, d'une part sur un mécanisme de *ségrégation spatiale,* isolant et protégeant les zones mémoires utilisées par chaque application, et d'autre part, un mécanisme de *ségrégation temporelle* qui garantit un séquencement des applications suivant un plan de charge déterminé à l'avance ;
- un *partage des moyens de communication* sur le réseau AFDX qui, grâce au concept de *Virtual Link*, permet une *ségrégation de la bande passante* et rend ainsi les communications déterministes ;
- *la norme avionique ARINC 653* [ARINC 2003] qui définit un standard d'interface commun pour toutes les applications développées dans le cadre de l'IMA, et permet à un équipementier de développer une fonction avionique sans se soucier des ressources matérielles utilisées et des autres fonctions susceptibles d'utiliser les mêmes ressources.

Mais à ce jour, certaines fonctions très critiques restent encore en dehors du concept IMA. C'est notamment le cas des CDVE, visés par nos travaux, et qui conservent encore une architecture classique à base de calculateurs dédiés.

1.5.3.1.4 Bilan des solutions actuelles pour les CDVE

Les solutions actuelles, qui ont été très largement éprouvées et dont la sûreté de fonctionnement n'est plus à démontrer, se basent sur une architecture de commande centralisée de type commande-surveillance. Les technologies numériques y sont déjà largement présentes au niveau des calculateurs et des communications inter-calculateurs (comme dans toutes les autres fonctions avioniques hors CDVE). Mais par contre, entre les calculateurs, actionneurs et capteurs, les communications sont encore analogiques avec des liaisons directes entre ces équipements. En outre, les équipements terminaux (actionneurs et capteurs) sont essentiellement passifs, l'intelligence de traitement étant concentrée dans les calculateurs.

Ces solutions n'exploitent donc pas tous les avantages des communications numériques, désormais incontournables. Avec pour conséquence directe de rendre impossible de tirer profit de l'énorme potentiel que ne saurait tarder à offrir les microsystèmes et leur utilisation massive. C'est tout l'enjeu de deux axes d'évolution pour les futurs CDV.

1.5.3.2 Commandes de Vol du Futur (CVF)

1.5.3.2.1 Communications numériques

Remplacer toutes les liaisons analogiques directes calculateurs-actionneurs-capteurs par des communications numériques permettra désormais d'avoir, au niveau global de l'avion, des traitements numériques de bout en bout, de la consigne du pilote jusqu'aux actionneurs.

Au niveau des CDV, les bénéfices attendus sont multiples et corrélés :

1) Des gains en poids, volume, complexité et donc coût global du câblage sont évidents.
2) L'ajout de nouveaux abonnés dans le système de commande sera beaucoup plus aisé (ex. : par "simple connexion" sur le bus).

3) Les traitements applicatifs et/ou réseaux pourront (ou devront) être répartis.
4) De nouvelles alternatives de sûreté de fonctionnement des CVF seront possibles. En effet, ce type de communication utilise d'autres techniques de codage et d'échange des ordres et des retours. Par exemple, les récepteurs pourront désormais eux-mêmes vérifier l'intégrité des messages reçus. Plus généralement, le caractère *"fail-safe"* pourra s'envisager au niveau de chaque élément de la chaîne, et non plus au niveau de la chaîne globale. Pour Airbus, l'effet espéré est de rendre inutile la ségrégation des deux canaux de commande et de surveillance, principe qui est au cœur des actuels CDVE.

Mais surtout, **les communications numériques sont un préalable incontournable** à l'utilisation massive de microsystèmes, qui accroîtra les avantages cités et en produira d'autres.

1.5.3.2.2 Intégration de nappes de microsystèmes

Une évolution encore plus significative, mais à plus long terme, sera d'exploiter les avantages des microsystèmes (cf. § 5.1. et 5.2.). De multiples utilisations de microsystèmes sont déjà à l'étude : nappes de capteurs de contraintes pour surveiller le vieillissement de la structure de l'avion, nappes de microcapteurs thermiques pour la détection de feu dans un réacteur, etc.

Pour les CDV, les microsystèmes envisagés sont principalement des microsurfaces de contrôle et les microactionneurs associés, en complément ou en remplacement des surfaces actuelles Des projets tels que AWIATOR et AEROMEMS (cf. § 5.2.3.2) traitent en partie des possibilités et des avantages d'utiliser des réseaux de telles microsurfaces. Les tailles envisagées pour ces microsurfaces sont de l'ordre du mm^2, cm^2 ou dm^2, avec donc nécessairement aussi une réduction de leurs moyens d'activation, au point même, qu'en dessous d'une certaine taille, d'autres sources d'énergies et de techniques d'activation seront possibles, si ce n'est nécessaires. Les plus étudiés sont les techniques électromagnétiques [Judy & Muller 1997] et électrostatiques [Kimura *et al.* 1997], (cf. § 5.2.4.1). Citons simplement quelques uns des avantages attendus (détaillés au § 5.2.4.1).

- Le poids des surfaces, mais aussi des actionneurs, ainsi que leur volume et leurs besoins en énergie seront inférieurs à ceux d'aujourd'hui.
- Une plus grande précision et souplesse de manœuvrabilité de l'avion seront possibles [Fermigier 2004]. En effet, la gestion des forces aérodynamiques pourra être plus fine, voire avec des nouveautés, puisque le niveau de granularité étant beaucoup plus fin, la distribution des forces sera partiellement différente de celle d'aujourd'hui. De nouvelles lois de commande seront sûrement à développer.
- Une plus grande tolérance aux défaillances sera possible, vu le nombre élevé des microsurfaces utilisées : perdre une ou des microsurfaces aura une incidence moindre.

Mais, comme déjà dit, l'utilisation massive de microsystèmes va introduire de nouvelles problématiques et contraintes, ne serait-ce qu'au niveau du système de communication.

Au bilan, qu'il s'agisse de l'utilisation de communications numériques, ou encore plus de nappes de microsystèmes, il est nécessaire de repenser les solutions actuelles des CDV, et donc, par voie de conséquence, la sûreté de fonctionnement des CDV. Ces évolutions conduiront nécessairement à de nouvelles générations de CDV, et sont donc très significatives.

1.5.4 Objectifs et spécifications des travaux

Rappelons que notre objectif global est de développer un système de communication adapté à la commande de nappes de microsystèmes dans des SCC à dynamique lente (cf. § 1.2.2.2). L'intégration à long terme de microsystèmes dans les CVF, avec en préalable, à court terme, le passage à des communications numériques, est le cadre de nos travaux. Nous allons donc montrer maintenant comment se décline cet objectif dans ce cadre, tout en précisant que les techniques que nous proposons par la suite peuvent être appliquées à d'autres systèmes. Nous allons donc préciser et compléter les hypothèses prises et les spécifications fonctionnelles et non-fonctionnelles (sûreté de fonctionnement) pertinentes pour nos travaux.

1.5.4.1 Hypothèses sur les microsystèmes

Du fait de la multiplicité des paramètres d'un ensemble de microsystèmes qui impacte les communications **de l'application CVF**, et du fait de leur caractère extrêmement novateur, nous n'avons envisagé que des microsurfaces et leurs microactionneurs comme microsystèmes. Et pour l'instant, nous n'avons considéré que des microsurfaces non critiques : des microspoilers, qui doivent, par centaines ou milliers, remplacer la dizaine de spoilers existants actuellement (cas de l'Airbus A320). Il s'agit donc de microsystèmes, fixes à la fois en nombre (très élevé) et en localisation.

Or, concrètement, de telles microsurfaces ne sont pas du tout à maturité, ni même encore à disposition. Nous n'avons donc raisonné que sur le principe de leur existence, et non sur de réelles caractéristiques techniques chiffrées. En particulier, nous n'avons fait aucune considération sur les actionneurs et la technique d'activation (hydraulique, électrostatique ou électromagnétique), pas plus que sur l'ordre du facteur de réduction.

1.5.4.2 Spécifications fonctionnelles

La *période de rafraîchissement* des asservissements est une caractéristique fondamentale des SCC considérés. Typiquement, la période avec laquelle les calculateurs de commande rafraîchissent les asservissements des surfaces de contrôle est de 10 ms, ce qui est notamment le cas pour les spoilers. Cela vaut aussi bien pour les envois des consignes (c'est-à-dire, les vitesses de braquage) des calculateurs aux actionneurs de ces surfaces, que pour les envois des retours de position de ces surfaces via des capteurs.

La *taille des messages* nécessaire pour véhiculer les informations de commandes et de retours de position est une autre caractéristique importante. L'identification de ces informations montre que quelques dizaines de bits de données utiles sont suffisants pour les représenter. Pour tenir compte des autres champs de ces messages (identifiant, adresse, etc.), nous prendrons une longueur de 100 bits utiles comme hypothèse de travail, étant précisé que cette valeur ne tient pas encore compte des bits de contrôle de la non-altération des données (ce point sera développé au chapitre 2).

Les deux informations précédentes permettent de déterminer *l'ordre de grandeur du débit*, que nous qualifierons "d'utile", et que doit garantir le système de communication pour assurer un rafraîchissement donné. Pour **z** surfaces de contrôle rafraîchies tous les 10 ms, et à raison d'une consigne et d'un retour de position par asservissement, cela donne 2 * **z** messages échangés toutes les 10 ms. Cela nécessite un débit utile de 200 Kbits/s pour commander et contrôler les 10 spoilers actuels.

1.5.4.3 Spécifications non-fonctionnelles et événement redouté

Le nouveau système de communication doit permettre d'assurer une sûreté de fonctionnement des CVF au moins équivalente à celle des CDV actuels. Pour notre part, cela signifie qu'il doit essentiellement satisfaire des contraintes de sécurité-innocuité (intégrité) et de disponibilité. Ces considérations se généralisent aux autres types d'applications concernés par nos travaux.

Il faudra tenir compte des considérations suivantes.

1. Nous ne traiterons que de la *sécurité-innocuité*, et pas de la *disponibilité*, pour laquelle nous dirons simplement qu'elle est assurée grâce à d'autres voies de commande sur lesquelles un basculement s'effectue lors d'un *défaut persistant* sur une voie.
2. Nous rappelons que les messages émis par les calculateurs de commande sont supposés être corrects, sachant qu'il est prévu que le principe de commande-surveillance exposé au § 1.5.3.1.2 s'applique jusqu'aux contrôleurs de réseau de ces calculateurs.
3. Enfin, comme il est exclu d'utiliser ce principe pour les autres éléments du réseau de communication, cela signifie que toute solution de redondance au niveau du système de communication est donc à rejeter (ex. : doublement des voies de communication).

En définitive nous ne traiterons que des scénarii susceptibles de conduire à l'événement redouté et liés à des défaillances pouvant altérer les messages véhiculés par le système de communication.

Bien que nous raisonnions sur les spoilers, et qu'un spoiler seul n'est pas considéré comme une surface critique (cf. § 1.5.2.2.2), le système de communication commandant un ensemble des spoilers est lui critique ; il fait partie des systèmes de classe A (cf. § 1.2.1). Cela veut dire que le taux de l'événement redouté (catastrophique) doit être inférieur à 10^{-9}/heure.

Dans les CDV, **l'événement redouté** est le fait d'appliquer des ordres erronés sur les actionneurs des surfaces de contrôle pendant un temps tel que cela se traduit par un braquage intempestif d'une partie des surfaces (appelé aussi embarquement de surface) (cf. § 1.5.2.2.2) :

1. qui ne pourrait pas être compensé (ou du moins pas assez rapidement) par les autres surfaces et amènerait l'avion dans une position catastrophique,
2. qui provoquerait des efforts excessifs sur la structure de l'avion (avec risque de rupture), du fait des efforts aérodynamiques induits ; cela est plus particulièrement vrai au niveau des attaches des empennages horizontaux et verticaux.

Typiquement, le premier cas n'est problématique que pour des embarquements excédant les 500 ms, alors que le second l'est pour des embarquements de l'ordre de 100 ms (temps suffisant pour atteindre 5° d'erreur de braquage avec des servocommandes permettant une vitesse maximale de 50°/s). C'est ce second cas qui est le plus contraignant. Nous considèrerons donc que l'événement redouté est un embarquement d'une durée de 100 ms.

La traduction au niveau du système de communication est donc la suivante (cf. § 1.2.3) : pour une surface de contrôle rafraîchie toutes les 10 ms, l'occurrence de l'événement redouté correspond au traitement de 10 consignes consécutives incorrectes et non détectées comme telles (d'où le fait qu'elles soient traitées comme étant des consignes correctes). Compte tenu des stratégies de recouvrement d'erreur utilisées, développées au § 2.2.4, nous considérerons, qu'en définitive, l'embarquement peut résulter de la non détection de 3 messages erronés parmi 10. De ce fait, le niveau d'intégrité du système de communication devra être tel que, le taux de cette non détection soit inférieur à 10^{-9}/heure.

1.6 Conclusion

Nos travaux sont fondés sur un double constat : les microsystèmes représentent de tels enjeux (techniques, économiques et sociétaux) que leur introduction massive dans de multiples applications est incontournable, mais par ailleurs, ce domaine est encore jeune et donc avec encore de nombreux défis à relever.

Dans ce contexte, notre objectif est la définition de systèmes de communication à haut niveau d'intégrité, adaptés à la commande de grands ensembles de microsystèmes. Le cas d'étude est une application dans le domaine de l'avionique : les systèmes de commandes de vol du futur, en vue de pouvoir commander des nappes de plusieurs centaines ou milliers de microsurfaces non critiques de type microspoilers.

Ce premier chapitre a présenté tous les éléments du cadre de nos travaux : une synthèse des enjeux et des défis des microsystèmes, les systèmes considérés pour leur intégration, les définitions générales de la sûreté de fonctionnement, puis plus précisément de l'intégrité des communications, et pour terminer, notre cas d'étude, et nos hypothèses et spécifications fonctionnelles et non fonctionnelles pour nos travaux.

Les CVF, qui sont le point de départ et le fil directeur de nos travaux, sont représentatifs de la classe de systèmes que nous considérons : les systèmes de commande-contrôle temps réel et critiques, à dynamique lente (telle que nous l'avons définie). Cette dernière caractéristique fait, qu'en termes de sûreté de fonctionnent, on cherche à se protéger d'un *lot d'occurrence de défaillances*, ce qui se traduit au niveau du réseau de communication par le fait que c'est la ***répétition de la non-protection*** vis-à-vis d'altérations des messages échangés entre les nœuds du réseau qui présente un risque. Et s'agissant de systèmes critiques, le taux d'occurrence d'un tel événement catastrophique doit être inférieur à 10^{-9} /h.

Cependant, les CVF ont pour spécificité de nécessiter, pour s'attaquer de manière approfondie à l'objectif visé, de traiter d'abord le problème du remplacement des communications analogiques actuelles par des communications numériques, qui sont un préalable incontournable. C'est pourquoi les microsystèmes ne sont présentés que partiellement dans ce chapitre, puis repris dans le chapitre 5. Néanmoins, l'étude préalable du passage à des communications numériques a pour avantage de pouvoir prendre en compte la future intégration de microsystèmes dès la phase de spécification du système de communication.

Et à ce stade de nos travaux, nous n'avons, dans un premier temps, pris en compte comme contraintes que le nombre élevé et le caractère fixe de ces microsystèmes (invariants en nombre et en localisation), et des microsystèmes qui ne communiquent pas entre eux. Et il faut ajouter que, de plus, le type de microsystème envisagé n'existe pas encore à l'heure actuelle.

Chapitre 2 - Système de communication envisagé et études des risques

Nous présentons maintenant dans ce chapitre les caractéristiques de base et les premières évaluations du nouveau système de communication envisagé pour répondre aux besoins de SCC à dynamique lente qui ont à commander des nappes de microsystèmes. Cette présentation s'appuie largement sur le cas des commandes de vol du futur. Et rappelons que notre préoccupation majeure est de mettre en place des solutions de communication à haut niveau d'intégrité, c'est-à-dire compatible avec un taux d'occurrence de l'événement redouté inférieur à 10^{-9}/h, et cela, sans faire appel à des redondances massives, et sachant que pour des SCC à dynamique lente, on cherche à se protéger contre la *répétition de la non-protection* vis-à-vis des altérations des messages échangés.

Nous commençons par présenter les caractéristiques de l'architecture de communication. Mais, nous n'en donnons que les éléments qui sont nécessaires aux analyses rapportées dans ce chapitre, analyses qui sont motivées, comme nous le montrerons, par le fait que cette architecture, à base de bus, comporte nécessairement des nœuds intermédiaires (d'interconnexions) actifs, notamment, du fait du nombre élevé de microsystèmes à abonner au réseau. Des développements plus précis sur l'architecture seront donnés dans le chapitre 5. Ensuite, nous présentons les aspects protocolaires et les stratégies de recouvrement d'erreur envisagés, fondés sur une étude des protocoles standard les plus largement utilisés dans les réseaux locaux, en s'attachant plus particulièrement à leurs techniques de détections d'erreurs.

Ces bases étant posées, il sera alors possible d'identifier les risques d'altération de l'intégrité des messages, associés aux différents éléments intervenant dans le système de communication, dont en particulier les risques associés aux nœuds intermédiaires actifs. Puis, sur la base des risques identifiés, nous déduisons les différents types de fautes à prendre en compte, et nous déterminons les caractéristiques des modèles d'erreur induits à considérer.

Enfin, en partant de ces caractéristiques et en prenant des taux d'occurrence usuels, nous évaluons la couverture de ces risques par des mécanismes de détection d'erreurs standard, pour déterminer si ces mécanismes permettent de satisfaire les objectifs d'intégrité fixés, ou si des compléments ou de nouvelles solutions sont à développer. Par exemple, pour notre cas d'étude, nous avons évalué le taux de non détection de 3 messages erronés dans un lot de 10 messages.

2.1 Système de communication envisagé

Cette partie décrit le système de communication envisagé pour un SCC. Nous traitons des aspects d'architecture, puis de protocoles de communication, plus particulièrement dans l'optique de l'intégrité des communications.

Le choix de l'architecture et des protocoles d'un réseau de communication dépend des besoins de l'application qu'il supporte et des contraintes à satisfaire qui en découlent. Nous focalisons surtout sur les contraintes temporelles et d'intégrité des communications [Kopetz & Bauer 2003], puisque nous considérons des SCC critiques qui, dans un délai fini, ont à collecter des informations de capteurs pour commander des actionneurs (élaborer et envoyer des commandes). Le délai d'exécution d'un tel cycle dépend à la fois du nombre et de la complexité des opérations à exécuter, et de la localisation des arguments nécessaires à cette exécution. Dit autrement, sur ce dernier point, plus les capteurs et les actionneurs sont nombreux et délocalisés, plus le temps d'exécution (traitement, asservissement) sera long. Cet aspect devient crucial dans le cas des nappes de microsystèmes.

2.1.1 Architecture

La commande de grands ensembles de microsystèmes est un domaine encore jeune. On constate que la plupart des études et réalisations sont pour l'instant construites autour d'architectures classiques centralisées. Or, ce type d'architecture atteint ses limites au-delà d'un certain nombre d'abonnés. Ce type n'est plus envisageable lorsqu'il s'agit d'échantillonner et de transmettre des commandes à des dizaines de milliers de microsystèmes, tant en termes d'architecture réseau que du point de vue de la charge de calcul induite pour le calculateur.

La distribution des traitements réseau, voire applicatifs, est donc à envisager. Dans la suite de ce chapitre, nous ne présentons que des grandes lignes sur nos choix architecturaux, les aspects distribution étant davantage développés dans le chapitre 5 (cf. § 5.3).

2.1.1.1 Architecture physique : à base de bus interconnectés

Pour les SCC usuels, les architectures réseau sont classiquement à base de bus. Or, de toute évidence, tous les abonnés ne peuvent pas être connectés sur un seul bus. Les raisons en sont :

1) le grand nombre d'abonnés (microsystèmes) envisagé, et la charge de traitement induite,
2) la robustesse requise par l'application en cas, soit de défaillances du bus (ex. : une rupture), soit de défaillances d'abonnés, qui amènent souvent à mettre en place des solutions à base de duplication et/ou de ségrégation (cf. § 1.5.3.1),
3) la facilité de mettre en œuvre une commande plus fine.

Donc, il est nécessaire d'utiliser plusieurs bus interconnectés par des nœuds intermédiaires. Rappelons que pour la partie de la robustesse sur laquelle nous focalisons – l'intégrité des communications – nous excluons les solutions faisant appel à de la duplication-redondance. Ces aspects ne sont pas à négliger, mais sortent du cadre de nos travaux.

2.1.1.2 Complexité et fonctionnalités des nœuds d'interconnexion

Pour répondre aux contraintes ci-dessus, et garantir les propriétés d'intégrité que nous avons retenues (cf. § 1.4.1.3), les nœuds intermédiaires à utiliser ne peuvent pas être que de simples répéteurs, qui par définition n'assurent que de la répétition, de l'adaptation ou de l'isolement galvanique de signal.

Des nœuds assurant au moins de la commutation sont nécessaires, si ce n'est d'autres fonctionnalités réseau plus complexes. La question est : lesquelles ?

Pour répondre exactement, il faudrait déjà connaître précisément le niveau “d'intelligence” des abonnés. Or ce n'est pas le cas. À ce stade de nos travaux, les microsystèmes envisagés ne sont pas encore choisis. Ensuite, même en supposant pouvoir définir le type de nœuds intermédiaires, la grande diversité de mise en œuvre d'un même type rend très difficile une analyse exhaustive de leur fiabilité et des risques qu'ils présentent. Et en outre, une telle analyse nécessite de connaître des taux précis chiffrés de défaillance, ce qui est d'autant plus difficile à obtenir que la complexité fonctionnelle de ces nœuds intermédiaires est grande.

En définitive, nos travaux ne s'appuient pas sur un ou des modèles fonctionnels précis de ces nœuds, mais sur la prise en compte d'un facteur commun à tous les types de nœuds possibles, (autres que de simples répéteurs) : ils possèdent tous au moins des capacités de mémorisation, voire de traitement. Notre analyse de risques s'appuie sur ces caractéristiques génériques.

2.1.1.3 Architecture logique

Pour pouvoir commander et asservir correctement des nappes de microsystèmes, il va falloir distribuer en partie la charge de calcul qu'aurait eu à supporter le calculateur dans une architecture centralisée le reliant directement à tous les actionneurs et capteurs [Lyshevki 2001]. Cette répartition concerne inévitablement au moins les fonctionnalités réseau, de par l'utilisation de nœuds intermédiaires actifs. Mais il se peut également qu'il soit nécessaire, ou mieux, opportuniste, de distribuer une partie des fonctionnalités applicatives (la commande).

Dans tous les cas, une architecture logique complexe de communication est à mettre en place avec comme problème de définir comment et selon quels critères répartir cette charge. Notamment, peut-on ou doit-on choisir une gestion répartie maillée (ou coopérative) dans laquelle une commande émanant du calculateur transite intégralement dans un réseau pour aboutir sur les actionneurs ? Ou alors, doit-on envisager une gestion distribuée de manière arborescente (ou hiérarchique) dans laquelle la commande issue du calculateur est d'un niveau d'abstraction différent de celui de la commande arrivant en définitive sur les actionneurs ?

Bien que ces aspects de répartition soient fondamentaux, à ce stade de la présentation de nos travaux nous précisons seulement que, s'agissant de SCC, nous avons opté pour une solution hiérarchisée. Nous ne débâtons pas ici plus avant de ce choix, mais seulement au chapitre 5 (cf. § 5.3). En effet, la prise en compte de l'influence des nœuds intermédiaires sur l'intégrité des communications, indépendamment d'une architecture précise, pose à elle seule suffisamment de problèmes, qu'il faut d'abord résoudre.

2.1.1.4 Architecture envisagée

Le principe de l'architecture que nous avons retenu est illustré sur la figure 2.1. Un calculateur commande les nappes de microactionneurs et acquiert les valeurs de microcapteurs par plusieurs bus de communication interconnectés pour former plusieurs niveaux de communication intermédiaires, organisé de manière hiérarchisée (ou arborescente).

De ce fait, les fonctionnalités réseau des nœuds intermédiaires peuvent différer d'un niveau à un autre. Comme, par exemple, être de moins en moins complexe à chaque fois qu'on descend d'un niveau vers les actionneurs, puisque le nombre de microsystèmes concernés par la commande du calculateur diminue.

Mais dans tous les cas, une partie au moins des nœuds intermédiaires sont actifs, c'est-à-dire avec au moins des capacités de mémorisation, voire de traitement des données qui y transitent.

On peut également envisager que les fonctionnalités applicatives des nœuds intermédiaires diffèrent selon leur nombre et leur niveau dans l'architecture de communication. Concrètement, par exemple, le calculateur adresse la commande aux nœuds intermédiaires de premier niveau auxquels il est directement connecté, qui à leur tour mémorisent la commande reçue et la traitent localement selon la nature et la valeur de l'information issue d'autres nœuds intermédiaires (soit du second niveau, soit de même niveau). La commande ainsi calculée localement par un nœud intermédiaire est envoyée à un ou plusieurs nœuds intermédiaires de second niveau et ainsi de suite jusqu'aux nappes de microsystèmes.

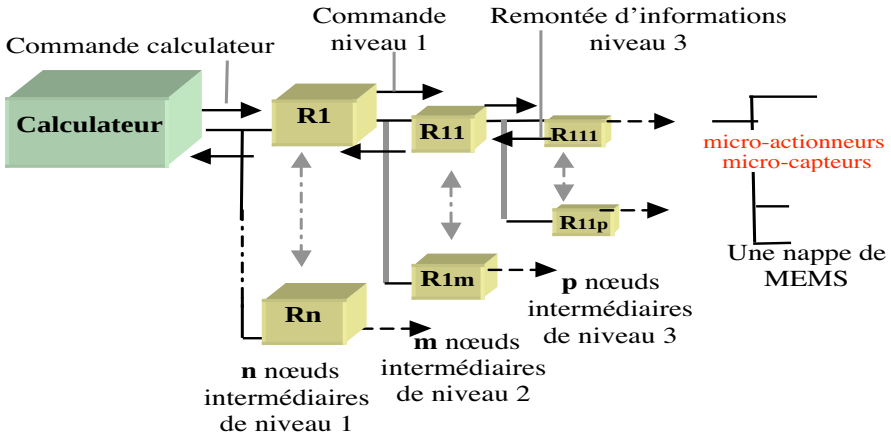


Figure 2.1 - Architecture de base

2.1.2 Protocoles

Tout comme l'architecture, le choix d'un protocole de communication doit répondre à plusieurs objectifs vis-à-vis du système auquel il est destiné (sûreté de fonctionnement, performances en termes de temps de latence, débit utile, etc.). Donc, choisir un protocole est un problème complexe. Cette complexité est renforcée par la diversité des protocoles existants, et ce d'autant plus que nous n'avons pour l'instant fixé que les bases de l'architecture, et non une architecture précise. Nous ne pouvons pas encore présenter ici des choix précis de protocoles, mais seulement des éléments pour guider ultérieurement ce choix.

2.1.2.1 Démarche générale

D'une manière générale, la démarche pour choisir un protocole suit les étapes suivantes :

1) Définir les besoins à satisfaire, pour en déduire les fonctions protocolaires nécessaires.
2) Chercher alors dans l'existant, s'il y a déjà des fonctions correspondantes. Or, déterminer le degré de correspondance n'est pas toujours trivial et nécessite souvent, pour être précisé, d'évaluer la réponse effective de ces fonctions aux besoins.
3) Si cette première phase d'évaluation n'est pas concluante, on en déduit les besoins de modifier ou de compléter des fonctions existantes, voire, de définir de nouvelles fonctions.
4) Et il faut alors entreprendre une seconde phase d'évaluation.

Toute évaluation doit se faire par rapport à des objectifs précis. Sachant que pour nous, la priorité est d'assurer l'intégrité des communications, cela nous amènera donc à identifier des risques à couvrir (cf. § 2.3) avant de pouvoir faire des évaluations. Rappelons également qu'il est naturel de commencer par explorer l'existant, pour des raisons d'économie et de confiance : développement, évaluation de performances et fiabilité déjà largement éprouvé.

En conséquence, cette partie sur les protocoles s'attachera surtout à présenter les besoins des systèmes considérés et leurs conséquences sur les fonctions protocolaires, suivi d'un état de l'art restreint aux protocoles jugés les plus significatifs.

2.1.2.2 Besoins protocolaires pour les SCC considérés

En se référant à l'architecture de communication de base présentée au § 2.1.1.4, ces protocoles de communication dépendent du volume et de la criticité des communications entre les nœuds d'un même niveau ou de niveaux différents, mais aussi, des contraintes temporelles à satisfaire, généralement fortes [Kopetz & Ochsenreitezr 1987]. La question d'un protocole pour des communications avec des nappes de microsystèmes est donc une question totalement ouverte.

Dans le chapitre 1, au § 1.5.4, nous avons précisé, au travers de notre cas d'étude, que s'agissant de SCC critiques à dynamique lente, les messages échangés sont des messages d'état, plutôt courts (ils véhiculent des ordres et des retours, qui peuvent même être booléens) : typiquement, une centaine de bits. Mais des débits élevés sont nécessaires, car :

- le nombre de messages peut être important (selon le nombre de microsystèmes),
- les commandes sont à rafraîchir, avec une forte contrainte de temps (quelques ms), d'autant plus difficile à satisfaire que le nombre de messages sera important.

Ajoutons que le protocole doit supporter de la diffusion, du *multicast* et de *l'unicast*.

Par ailleurs, de par la complexité d'une partie des nœuds d'interconnexion, le réseau de communication comportera plusieurs couches au sens du modèle de référence OSI. Et, de ce fait, pour ce qui est des contraintes d'intégrité des données transmises de bout en bout du système de communication, elles sont considérées suivant trois niveaux :

- physique : le support physique et la structure interne des microsystèmes,
- réseau : les modes de communication et les protocoles associés,
- application ou traitement : l'architecture générale de la répartition du traitement.

2.1.2.3 État de l'art restreint

Avec les considérations faites jusqu'ici, nous avons entrepris une étude des protocoles existants pour la couche 2 (toujours en référence au modèle OSI), et les plus largement utilisés dans des SCC. Nous avons considéré des critères techniques (bande passante, nombre maximum de nœuds supportés, méthodes d'accès au bus, technique de codage des bits) et des critères de fiabilité (techniques de contrôle d'erreur, possibilités d'utiliser plusieurs modes et d'évaluer le protocole). Pour la fiabilité, nous avons limité notre étude aux protocoles qui n'utilisent que des techniques de détection d'erreurs, sans correction au niveau protocolaire. La "correction", qui est en fait du recouvrement d'erreur, est laissé à la charge de l'application et sera développée aux § 2.2.4 et 2.2.5. Dans ce cadre, nous avons donc analysé les protocoles standard, dont principalement CAN, TTCAN, TTP/A et TTP/C. En effet, ils sont représentatifs de deux familles de protocoles de communication que l'on peut distinguer selon leur principe de contrôle d'accès au bus de communication : les protocoles *asynchrones* (ou ETP : Event Triggered Protocol) et les protocoles *synchrones* (ou TTP : *Time Triggered Protocol*).

Le principe des protocoles asynchrones est l'émission des messages à des dates d'occurrence d'événements spécifiques (bus libre, réception d'une requête pour l'émission). Les délais de transmission sont variables et non bornés d'où une prédictibilité limitée du système. Par contre, ces protocoles peuvent assurer un ordonnancement dynamique qui favorise la flexibilité du système, puisqu'il est aisément possible de modifier ou d'ajouter des tâches sans avoir à toucher à l'ensemble des nœuds du réseau.

Quant aux protocoles synchrones, leur principe est l'émission des messages à des dates de cycle spécifiques prédéfinies (*time slot*). Les délais de transmission sont bornés donc connus, d'où une forte prédictibilité du système. Par contre, l'ordonnancement est statique, puisque toute modification ou ajout de tâches impose de retoucher à l'ensemble des nœuds du réseau.

Le protocole TTP/C [Kopetz 1997] est le plus représentatif de la famille des protocoles synchrones. C'est une version complète, tolérante aux fautes et à contrainte temporelle forte [TTTECH 1999]. La structure de ce protocole, ainsi que son mécanisme de synchronisation distribuée d'horloges, assurent une faible gigue. Ce protocole est robuste, vu qu'un nœud peut céder l'exécution de sa tâche à un autre en cas de défaillance. En plus, TTP/C résout le problème de *babbling idiot* (existant dans CAN) grâce à la technique du bus guardian qui gère l'autorisation d'accès en écriture d'un nœud au bus [Bannatyne 1999]. Le *babbling idiot* est le fait qu'un nœud du réseau essaye d'envoyer répétitivement un message alors qu'il n'en a pas le droit, ce qui peut causer la perte totale des communications dans un bus à simple voie.

2.1.2.4 Premières conclusions sur les protocoles

De l'étude détaillée effective que nous avons menée sur les protocoles existants [Youssef *et al.* 2003a], nous n'avons gardé ici que les éléments strictement essentiels à la poursuite de la présentation de nos travaux dans ce mémoire.

En effet, comme nous le verrons dans la suite de ce chapitre, il s'avère qu'après une étude de couverture des risques vis-à-vis de l'intégrité, tous ces protocoles présentent des insuffisances, qui nous ont amené à proposer une solution qui en définitive fait abstraction du choix d'un de ces protocoles.

Cependant, pour mieux illustrer ces constats à venir, nous donnerons, sur notre cas d'application des CVF, un exemple concret du protocole à privilégier. En effet, il faut simplement retenir que les conditions identifiées dans notre travail et qui recherchent davantage la fiabilité d'un protocole que sa flexibilité nous conduit à conclure, tout comme dans [Rushby 2003] et [Navet *et al.* 2005], qu'il faut privilégier des protocoles de type synchrone, avec un intérêt plus particulier pour le protocole TTP/C.

2.2 Illustration du système de communication : cas des CVF

Nous allons maintenant nous appuyer sur notre cas d'étude, les CVF, à la fois pour illustrer plus précisément les principes généraux retenus ci-dessus pour le système de communication, et également, pour développer nos considérations sur les techniques de détection d'erreurs et les stratégies applicatives de recouvrement associées.

Ce qui nous permettra par la suite de mener des analyses par rapport à des objectifs d'intégrité et de temps réel. Et bien que ces objectifs soient liés aux CVF, ils sont représentatifs d'un grand nombre de SCC critiques et temps réel.

2.2.1 Architecture de communication

La figure 2.2 montre de manière simplifiée, l'application du principe de l'architecture décrite ci avant, à l'exemple d'un Airbus A320 qui met en jeu 24 asservissements (2 servocommandes sur chacun des 2 ailerons, 1 servocommande sur chacun des 10 spoilers, 3 servocommandes pour la gouverne de direction, 2 servocommandes sur chacun des 2 élévateurs, 3 moteurs électriques de plan horizontal réglable). Pour des raisons de disponibilité et de ségrégation électrique, l'architecture de communication envisagée présentera 4 bus (2 pour chacun des 2 cotés électriques ségrégés), soit 6 asservissements par bus.

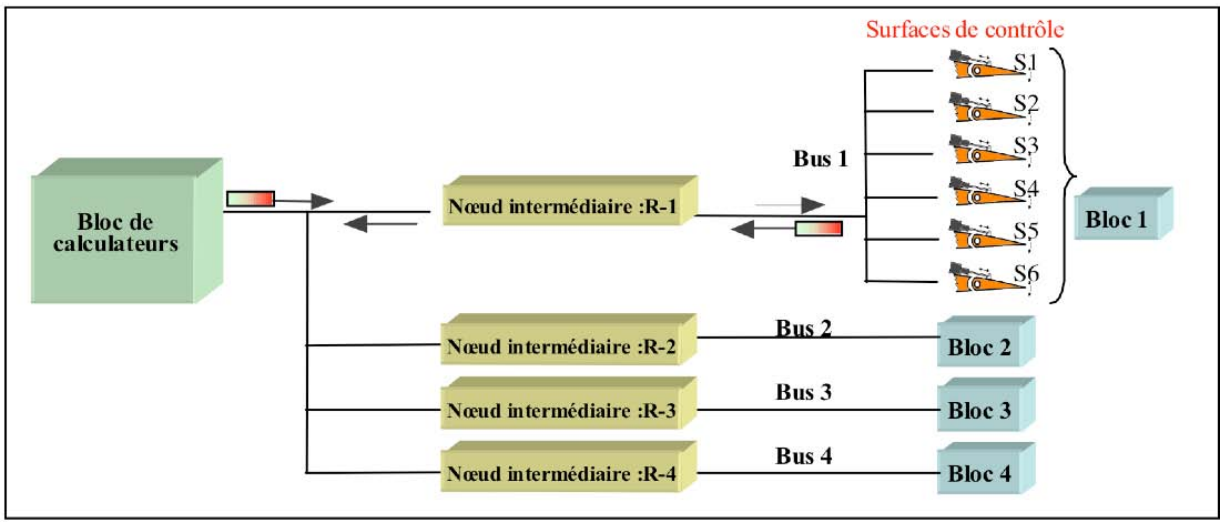


Figure 2.2 - Architecture simplifiée de communication pour les CDV d'un Airbus A320

2.2.2 Protocole synchrone : constitution du cycle de base

Sur la base de cet exemple d'architecture, nous donnons une illustration d'un exemple d'utilisation d'un protocole de type TTP/C, puisque le cycle de base d'un tel protocole dépend de l'architecture du système auquel il est appliqué.

La définition du cycle de base doit prendre en compte la contrainte temporelle fixée par les spécifications (cf. § 1.5.4.2) : chaque surface de contrôle doit être rafraîchie toutes les 10 ms. À raison d'une consigne et d'un retour de position par asservissement, et de 6 asservissements par bus, 12 messages doivent être échangés sur le bus toutes les 10 ms. En comptant le message d'initialisation diffusé aux différents nœuds du système de communication, 13 messages en tout sont échangés durant les 10 ms, ce qui correspond à 13 fenêtres temporelles. Dans ce cas, et en faisant une allocation équiprobable de ces fenêtres, chacune aura une largeur de (10/13) ms $\approx$ 0.77 ms.

Le tableau 2.1 présente, sur un cycle de base qui s'étend sur 10 ms, un ordonnancement statique du système pour un des 4 bus de communication. La phase de synchronisation qui dure 0.77 ms = 1 UT (Unité Temporelle) est nécessaire au début du cycle de base. Elle est basée sur la réception par tous les nœuds du système d'un message d'initialisation émis par un nœud spécifique (*Time Master Node*) [Pfeifer *et al.* 1999]. Ces messages permettent la synchronisation entre les différentes entités du système, car ils contiennent un champ qui, suivant les caractéristiques du protocole, renferme, soit le temps global, soit le temps local du système. La différence entre ces deux paramètres est que le temps local du système s'initialise à chaque début d'un cycle de base [Rushby 2003].

Entité	*Temps déclenchant (ms)*	*Latence de communication (ms)*
Synchronisation	s'étend sur 0.77 ms au maximum	
R	0.77	0.77
S1	1.54	0.77
R	2.31	0.77
S2	3.08	0.77
R	3.85	0.77
S3	4.62	0.77
R	5.39	0.77
S4	6.16	0.77
R	6.93	0.77
S5	7.7	0.77
R	8.47	0.77
S6	9.24	0.77

Tableau 2.1 - Exemple d'ordonnancement statique sur un cycle de base

Après cette phase de synchronisation, un nœud intermédiaire R ré-amplifie la consigne numérique émise par le calculateur, décode éventuellement les données, et les retransmet à 1 UT vers la servocommande S1. Le retour de position par la servocommande est émis à 2 UT. Ce processus s'étend sur 1.54 ms et concerne l'asservissement d'une seule surface de contrôle parmi les 6. Il se répète à 2.31, 3.85, 5.39, 6.93 et 8.47 ms respectivement pour les servocommandes S2, S3, S4, S5 et S6. (cf. figure 2.3).

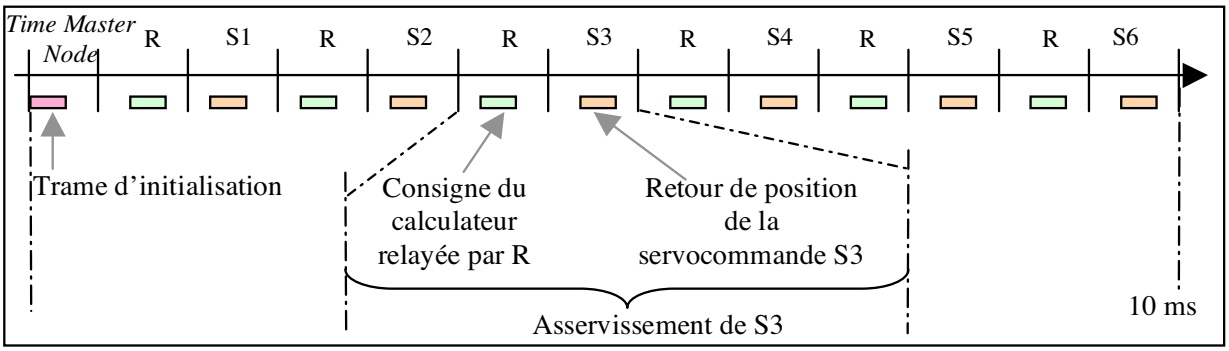


Figure 2.3 - Cycle de base du système de communication

2.2.3 Techniques de détection des erreurs de transmission

Examinons maintenant les perturbations qui peuvent inévitablement altérer les transmissions, et ce, quelle que soit la qualité des supports de communication et les performances des techniques de transmission utilisées. Nous distinguons trois types de situations :

1) message reçu mais avec un contenu altéré, où les altérations se traduisent, comme vu au chapitre 1, par des erreurs en valeurs, ou temporelles, ou d'adressage,
2) message perdu pendant sa transmission dans le réseau de communication,
3) messages arrivant dans le désordre.

Dans le cadre de la thèse, nous nous intéressons uniquement au premier cas, dans lequel la suite binaire reçue n'est pas identique à la suite émise (altération des données transmises). En effet, pour les applications considérées, le volume de l'information utile (commande ou retour) ne nécessite pas plusieurs messages pour être transmis.

Le problème de désordre des messages ne se pose donc pas. Quant à la perte de message, nous admettons qu'elle est couverte par des stratégies de recouvrement présentées dans les sections suivantes. Qui plus est, il est évident qu'un message perdu ne peut pas être utilisé par le récepteur. Alors qu'un message erroné qui arrive jusqu'au récepteur sera consommé si on ne détecte pas qu'il est erroné. Ce qui signifie qu'une commande erronée sera appliquée, avec potentiellement de graves conséquences.

Nous avons montré au § 1.4.2 que pour se protéger de ces altérations, il fallait au moins pouvoir les détecter (via des codes détecteurs, par exemple), si ce n'est les corriger (via des codes correcteurs, par exemple), ou compléter la détection par des stratégies de recouvrement. Nous avons également spécifié que nos travaux s'intéressent uniquement aux techniques de détection, même si les codes correcteurs, comme les turbo-codes, sont de plus en plus utilisés, mais surtout pour se protéger des altérations sur des canaux très bruités comme les liaisons sans fil. En outre, dans les systèmes actuels de CDV, les moyens de détection reposent sur une architecture de type commande-surveillance, qui impose une duplication systématique des nœuds. Or une de nos spécifications est de chercher à utiliser des techniques de codage des informations (codes détecteurs d'erreurs) qui permettent d'éviter cette duplication autant que faire ce peut.

Nous avons donc mené une étude comparative détaillé [Youssef *et al*. 2003a], des différentes techniques de détection d'erreurs de transmission pour trouver celle qui offre le meilleur compromis entre capacités de détection et temps de génération et vérification des bits de contrôle. Nous en avons retenu les codes à redondance cyclique (CRC). Nous résumons ici notre étude, plus particulièrement dans le domaine des bus de terrain et avioniques, dans lequel seule la détection est généralement mise en œuvre.

La protection est très faible dans le cas des réseaux basés sur ARINC 429 exploités sur des avions civils récents (Airbus A310 à A340, Boeing 727 à 767, McDonnel Douglas MD-11), puisqu'elle n'est effectuée que par un simple bit de parité [ARINC 2002]. Il est en de même pour l'ARINC 629 développé au départ pour le Boeing 777 ou le MIL-STD-1553, et utilisé plus tard dans les systèmes avioniques militaires.

La protection est plus importante, avec l'utilisation de codes CRC, dans les bus avioniques plus récents comme ARINC 636 ou AFDX, qui ont été développés dans le cadre du concept de l'avionique modulaire intégrée (IMA). Les spécifications de l'ARINC 636 sont adaptées à partir de celles du protocole FDDI largement utilisé en informatique pour les épines dorsales (backbones) des réseaux locaux. L'AFDX est un sous-ensemble de l'ARINC 664 développé par Rockwell Collins pour Airbus qui est lui-même une adaptation des normes du réseau Internet (Ethernet, protocoles IP, UDP, etc.) pour une utilisation avionique. Ces bus concernent les liaisons inter-calculateurs dans un avion avec des longueurs de trames pouvant dépasser 1000 bits. Ils ne sont donc pas forcément représentatifs de ce qui pourrait être utilisé dans le cas de SCC où la longueur des trames est plutôt de l'ordre de la centaine de bits.

Les codes CRC 16 bits sont également couramment utilisés dans les protocoles de communication des systèmes industriels embarqués (CAN, TTCAN, TTP/C, etc.) [Castagnoli *et al*. 1990]. TTP/C, quant à lui, présente en plus une technique de codage flexible puisqu'il est compatible avec les CRC à 16 et 24 bits [Curtis & France 1999]. Cependant, TTP/C propose une duplication des liaisons pour une protection extrême.

D'où en définitive, pour les analyses qui vont suivre, nous considérons les codes CRC 16 bits qui nous semblent être les plus représentatifs de ce qui pourrait être retenu, en particulier pour les CVF.

2.2.4 Stratégies de recouvrement

Avant de mener ces analyses, il reste encore à présenter les stratégies de recouvrement d'erreur qui viennent compléter la détection d'erreurs. Ces stratégies mettent en œuvre une réaction en cas de détection d'erreurs : altération d'un message reçu, mais aussi perte d'un message. Elles sont à la charge de l'applicatif. Nous illustrons ce principe au travers de ce qui est fait dans les CDV.

Suite à la détection d'une erreur, une stratégie de recouvrement appropriée doit être mise en œuvre. Une telle stratégie résulte souvent d'un compromis, car elle doit tout à la fois :

1) garantir un rafraîchissement correct des commandes,
2) ne pas conclure trop rapidement à la défaillance de la voie de communication,
3) ne pas nuire au niveau de la sécurité innocuité.

Analysons tout d'abord avec quelle fréquence un recouvrement sera engagé. Cette fréquence est directement liée à la fréquence de détection d'une erreur, elle-même liée à la fréquence d'apparition d'une erreur. Les erreurs les plus fréquentes dans un réseau de communication sont celles liées au bruit sur le canal. À raison de 100 bits par message échangé et avec un BER (*Bit Error Rate*) de 10^{-8}/bit (taux courant pour un LAN), en moyenne 1 message sur 10^{6} est altéré et conduit à la détection d'une erreur. En raisonnant au niveau d'un asservissement, à raison de 2 messages par asservissement (commande et retour de position) et d'un rafraîchissement de chaque asservissement tous les 10 ms, cela correspond à un taux de détection d'erreurs sur chaque asservissement de : 10^{-6} . 2 . 100 Hz . 3600 s = 0,72 / heure.

Avec un tel taux, plusieurs détections d'erreurs peuvent intervenir sur un asservissement au cours d'un vol. Il n'est alors pas admissible que la détection d'une erreur sur une voie de communication conduise systématiquement à une commutation sur une voie de secours.

La stratégie de recouvrement retenue consiste en premier lieu, suite à la détection d'une erreur, à utiliser la consigne du cycle précédent, ce qui est tout à fait possible et logique compte tenu de la dynamique des systèmes considérés. C'est une stratégie couramment adoptée à ce jour dans les systèmes de contrôle, notamment avec les liaisons ARINC [ARINC 2002]. La stratégie consiste ensuite à ne déclarer défaillante la voie de communication, qu'à l'issue de plusieurs détections d'erreurs. Ce genre de filtrage est également d'une pratique courante.

D'un autre côté, il faut être conscient qu'attendre trop longtemps avant de déclarer défaillante la voie de communication peut présenter un risque vis-à-vis de la sécurité que nous évoquons dans le paragraphe suivant. Dans le chapitre 3, nous traiterons également de cet aspect en montrant le compromis à faire entre la disponibilité de la voie de communication et la sécurité du système commandé.

Dans le cadre de nos analyses, nous considérerons que la voie sera déclarée défaillante au bout de 3 détections d'erreurs et nous étudierons deux stratégies :

1) la voie est déclarée défaillante après la détection d'une erreur sur 3 messages successifs,
2) la voie est déclarée défaillante au bout de 3 détections d'erreurs dans un lot de 10 messages successifs.

2.2.5 Stratégie de recouvrement et embarquement d'une surface

La stratégie de recouvrement utilisée peut avoir un impact sur la probabilité d'occurrence de l'événement redouté qui, pour nous, est l'embarquement d'une surface de contrôle.

Ainsi, la stratégie qui consiste, d'une part, à utiliser la consigne précédente dans le cas de la détection d'une erreur sur un message, et d'autre part, à ne déclarer défaillante la voie de communication qu'au bout de 3 détections successives par exemple, peut, dans le cas d'un lot de 10 messages, conduire au scénario le plus pénalisant suivant :

non détection sur le 1er message, détection sur le 2^{e} et 3^{e} message mais pas sur le 4^{e}, détection sur le 5^{e} et le 6^{e} message, mais pas sur le 7^{e}, etc.

Cet enchaînement, qui conduit à appliquer une consigne erronée sur 10 cycles successifs, semble être fort improbable. Mais en réalité, il n'est pas si improbable que ça dans le cas de l'utilisation de la protection que nous proposons au chapitre 3. Il mérite donc d'être décrit, d'autant plus que nous y ferons référence par la suite.

Rappelons d'abord que la vitesse maximale de déploiement des servocommandes est de 50°/s. Si l'altération d'un message se traduit par exemple par un écart de 5° entre la consigne émise et celle reçue, et si le message erroné n'est pas détecté, alors ce message sera utilisé sur 10 cycles, soit 100 ms. Ce qui conduit à l'embarquement de la surface de contrôle. La figure 2.4 illustre ce scénario.

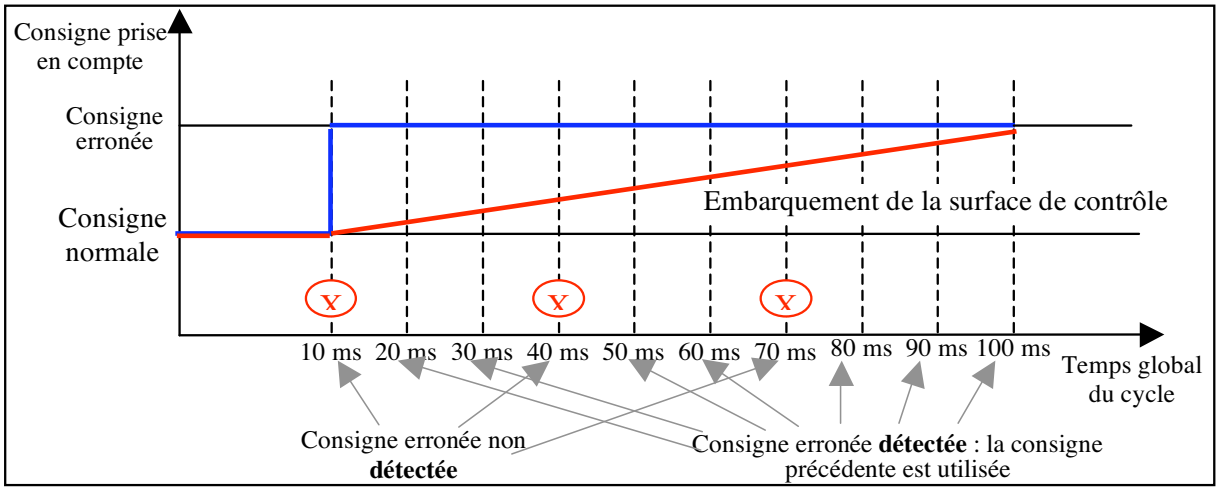


Figure 2.4 - Illustration de l'embarquement d'une surface de contrôle

En conclusion, dans les évaluations de risques qui suivent, nous prendrons comme objectif d'intégrité simple, mais conservateur, de rendre le taux de non-détection de 3 messages erronés sur un lot de 10 messages << 10^{-9}/h.

2.3 Identification des risques à couvrir

Il est maintenant possible d'identifier les risques à couvrir au niveau du système de communication, et dont il faut vérifier la couverture par la fonction de contrôle utilisée. À chaque type de risque (faute) est associé un taux d'occurrence, et suivant le type de risque, nous nous sommes attachés à préciser si celui-ci peut altérer de manière indépendante plusieurs messages successifs. Nous terminons cette section par la présentation des modèles d'erreurs associés à chaque type de faute considéré.

Le premier des risques à couvrir, et souvent le seul pris en compte, est lié au bruit intrinsèque des composants utilisés. Pour notre part, nous considérons également les risques associés aux défaillances du câblage ainsi qu'à celles des nœuds intermédiaires.

2.3.1 Risques associés au bruit

Le risque associé au bruit intrinsèque des composants est une inversion d'un bit (*bit-flip*) au niveau du message transmis sur le canal de communication. Compte tenu du phénomène mis en jeu, il est possible de considérer qu'il y a une indépendance des différents bits d'un message vis-à-vis de leur risque d'altération. Par conséquent, au niveau de plusieurs messages successifs, il y a également indépendance entre les altérations qui pourraient les affecter.

Le taux d'occurrence associé est le Bit Error Rate (BER). Par la suite, nous prendrons comme valeur le BER observé pour des LAN qui est de 10^{-8}/bit.

2.3.2 Risques associés aux défaillances du câblage

Un des risques associés au câblage est que celui-ci soit soumis à des perturbations externes (ex. : des perturbations électromagnétiques) qui pourraient induire des erreurs sur plusieurs bits successifs d'un message (erreurs en rafale). Si le nombre de bits successifs erronés est inférieur à la distance de Hamming du code détecteur d'erreurs utilisé, le risque est couvert par ce code.

Un autre risque associé au câblage est une rupture ou un court-circuit qui affecte de manière transitoire un connecteur ou un câble (ex. : faux contacts). Avec un tel phénomène, le nombre de bits d'un message pouvant être erronés peut être très important et plusieurs messages successifs peuvent être affectés. Toutefois, s'agissant d'un phénomène aléatoire, il n'y a aucune corrélation entre les erreurs affectant différents messages. En d'autres termes, la probabilité de détection d'un message erroné est tout à fait indépendante du fait qu'une défaillance du câblage se soit traduite par une non détection au niveau du message précédent.

Par la suite, nous prendrons un taux d'occurrence de défaillance du câblage égal à 10^{-5}/h.

2.3.3 Risques associés aux défaillances des nœuds intermédiaires

Les risques associés aux nœuds intermédiaires sont liés au type de fonctions que réalisent ces noeuds, qui peuvent aller de la simple amplification du signal (fonction d'un répéteur), à des manipulations des messages reçus, en passant par leur mémorisation et leur retransmission.

Lorsque la fonction d'un nœud intermédiaire se limite à une amplification du signal, ou une adaptation à un autre support physique, il n'est pas besoin de mémoriser le message. Car dans ce cas, le traitement intervient simplement au niveau du signal physique (le nœud intermédiaire est un simple répéteur). Les risques associés ne sont pas différents des risques associés à un défaut de câblage (cf. § 2.3.2).

Lorsque la fonction du noeud nécessite de mémoriser la totalité du message avant de le réémettre, les risques sont plus importants. En cas de défauts multiples (multiples bits collés à 0 ou à 1) sur les éléments de stockage du nœud [Blanc & Gil 2003], les erreurs peuvent être telles, que les risques ne sont pas couverts par la fonction de contrôle utilisée. Dans le cas où la fonction de contrôle est basée sur un code CRC, ce cas de figure correspond à une erreur qui serait multiple du polynôme générateur utilisé. Au niveau d'un seul message, les risques ne sont toutefois pas différents d'un défaut de câblage. Par contre, les risques sont différents si l'on raisonne sur plusieurs messages successifs. Les défauts étant permanents, il y a un risque que les erreurs soient répétitives sur plusieurs messages ; ça sera notamment le cas si les données ne changent pas d'un message à un autre.

Un contrôleur peut aussi présenter des risques vis-à-vis du rafraîchissement. Dans le cas où la zone mémoire ne serait plus accessible en écriture, il peut toujours émettre le même message.

Il peut aussi y avoir des risques de falsification du contenu d'un message dans le cas d'un nœud qui accède au bus de communication pour émettre au mauvais moment. Le message émis risque alors de se combiner avec un autre message, et donc de falsifier ce dernier, qui plus est, éventuellement de manière répétitive d'un cycle à l'autre si les données ne changent pas. Ces risques seront considérablement réduits si le contrôleur inclut un gardien de bus censé protéger le bus de communication contre les nœuds qui ont perdu leur synchronisation, ou qui sont devenus *babbling idiots* [Poledna *et al.* 2001].

Le risque extrême est associé au cas où le nœud intermédiaire a la possibilité de traiter le contenu d'un message. Dans ce cas, le contrôleur, par l'intermédiaire de la fonction de contrôle utilisée, génère de nouveaux bits de contrôle sur la base du message traité. Les erreurs pouvant ainsi intervenir pendant la phase de traitement du message ne sont pas couvertes.

2.3.4 Modèles associés d'erreur

Pour analyser les risques induits par les différents types de défaillance présentés précédemment, et voir comment ils sont couverts par des mécanismes de détection d'erreurs, il convient d'associer un modèle d'erreur à chaque type de faute. En plus de l'étendue spatiale de l'erreur induite (nombre de bits affectés), il est également important de considérer l'aspect temporel de l'erreur (erreur répétitive sur plusieurs messages successifs ou non) [Askerdal *et al.* 2002].

Le modèle d'erreur associé aux altérations induites par le bruit est celui d'*erreurs aléatoires indépendantes* d'un message à un autre. Il caractérise le fait que le nombre de bits altérés reste faible et qu'il n'y a pas de corrélation entre les bits affectés au sein d'un message, et par conséquent sur des messages différents. Le modèle d'erreur généralement associé aux altérations induites par des perturbations électromagnétiques sur le câblage est celui d'*erreurs en rafale (burst)*. Les bits affectés ne concernent qu'une partie du message et sont en paquet ; même si plusieurs messages successifs peuvent être affectés par les perturbations, le motif de l'erreur ne sera pas le même d'un message à un autre, et donc il s'agit d'*erreurs en rafale indépendantes*.

Ces deux premiers modèles d'erreur sont généralement les seuls retenus dans les documents relatifs au protocole Internet et les RFC le concernant [RFC 3385]. En ce qui nous concerne, nous considérons deux autres modèles d'erreur qui concernent, d'une part, les altérations dues à d'autres défauts de câblage tels les courts-circuits ou faux contacts, et d'autre part, les altérations apportées par les nœuds intermédiaires.

Pour les erreurs induites par des défauts de câblage, autres que les perturbations électromagnétiques, nous prenons des *erreurs multiples indépendantes* comme modèle d'erreur. Cela revient à considérer que n'importe quel bit d'un message peut être affecté, mais que les bits ne seront pas nécessairement affectés de la même façon d'un message à un autre. Dans le cas des nœuds intermédiaires, nous retenons le modèle d'erreur le plus pénalisant : celui *d'erreurs multiples répétitives*. Ce qui distingue ce modèle d'erreur du précédent est que la même erreur se reproduit sur plusieurs messages successifs.

2.4 Évaluations des risques

Dans cette section, nous allons évaluer dans quelle mesure les codes CRC, mis en œuvre classiquement dans les couches basses d'un réseau, permettent de faire face aux risques identifiés précédemment. Nous prendrons le cas des CRC 16 bits qui sont les plus couramment utilisés à ce jour. Bien que, pour des longueurs de message de 100 bits (cas envisagé pour notre cas d'étude), certains codes CRC 16 bits permettent d'obtenir une distance de Hamming de 6, nous envisageons par la suite le cas pessimiste correspondant à une distance de Hamming de 5. L'évaluation concerne non seulement le taux d'un seul message erroné non détecté, mais aussi le taux de 3 messages erronés et non détectés dans un lot de 10 messages qui, rappelons le, constitue l'événement redouté à considérer pour notre cas d'étude.

Mais avant de présenter les résultats de ces évaluations, il faut d'abord présenter le mécanisme de détection des erreurs qui a servi à illustrer nos propos.

2.4.1 Représentation du mécanisme de détection des erreurs

La figure 2.5 illustre la transmission d'un message de bout en bout du système de communication, à travers des couches protocolaires, par analogie à la représentation OSI.

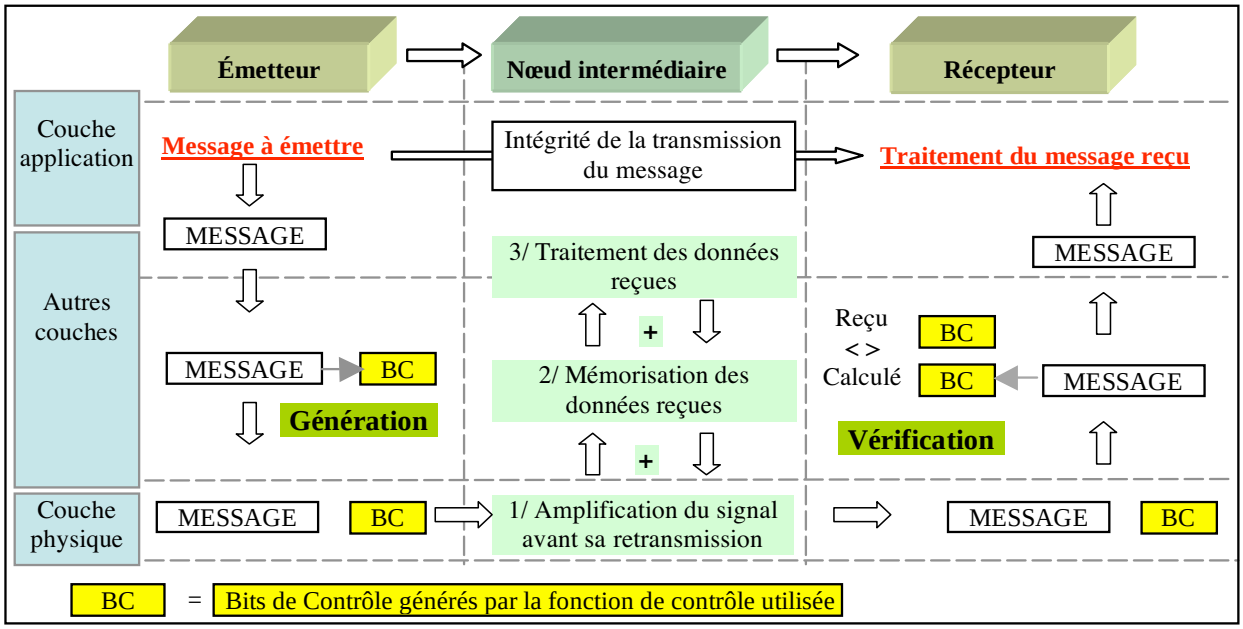


Figure 2.5 - Mécanisme de détection des altérations des données transmises

Cette représentation en couches reflète le fait que les données de la couche application du nœud émetteur sont transmises aux couches inférieures, dont chacune ajoute d'autres informations (champs) encapsulant ainsi les données de la couche du dessus.

Les champs ajoutés dépendent du protocole utilisé entre les nœuds émetteurs et récepteurs au niveau de la couche concernée. Côté récepteur, le processus inverse (décapsulation des données) s'opère au niveau des couches du nœud récepteur. À cette illustration de la transmission d'un message entre deux nœuds du système, s'ajoute la spécification des fonctions qui permettent de contrôler la non altération de ce message (**F**onctions de **C**ontrôle : **FC**). Les bits de contrôle (**BC**) générés par ces fonctions ont pour but de détecter les erreurs de transmission. Il s'agit des bits de CRC gérés matériellement par chacun des nœuds du système au niveau des couches basses.

La figure 2.5 illustre également les fonctionnalités du nœud intermédiaire. Les fonctions de base de cet élément sont la restauration du signal et l'adaptation éventuelle de supports de communication de types différents. La restauration concerne le cadencement, la forme et l'amplitude du signal reçu avant sa retransmission. Dans ce cas, il n'y a ni de mémorisation, ni de manipulation des informations, et ni de re-calcul des bits de contrôle. Dans le cas de nœuds complexes ou actifs, comme des routeurs, le nœud a besoin de mémoriser temporairement le message avant d'y appliquer un traitement réseau (modification de certains champs) impliquant un re-calcul des bits de contrôle. Dans un cas intermédiaire, le nœud peut avoir juste besoin de mémoriser le message, mais sans y appliquer de traitement.

2.4.2 Risques associés au bruit

Pour évaluer les risques associés au bruit, nous nous basons sur la formule fournie dans [Castagnoli *et al.* 1990], qui donne, pour un code de longueur n, la probabilité P_{ue} de non détection des messages erronés sur un canal binaire symétrique ayant un taux ε d'erreur bit (BER) dans l'intervalle [0, 0.5] :

$$\mathbf{P_{ue}(\varepsilon, n)} = \sum_{i=d}^{n} \mathbf{(A_i * (\varepsilon^i) * (1-\varepsilon)^{(n-i)})}$$

$$\mathbf{\sim= A_d * \varepsilon^d \quad (pour\ \varepsilon << 1)}$$

$$\mathbf{\sim= 2^{-p} - 2^{-n} \quad (pour\ \varepsilon = 0{,}5)}$$

avec : A_i le nombre de mots de code de poids i (c'est-à-dire avec i bits à 1), d la distance de Hamming du code, p le nombre de bits de contrôle.

Ainsi, la probabilité P d'avoir un message de 100 bits, qui soit erroné, et non détecté par un code CRC à 16 bits et d'une distance de Hamming de 5 est donnée, dans le cas d'un BER égal à 10^{-8}/bit, par :

$$P = (116! / 5!\ 111!) * (10^{-8}/\text{bit})^5 \text{ soit} \approx 10^{-32}/\text{message}$$

Dans le cas d'un système de communication présentant un débit de 1 Mbits/s, en prenant des messages de longueur 100 bits, on a un maximum de 10^4 messages/s, soit $3{,}6.10^7$ messages/h. Le taux de messages erronés non couverts par le CRC est alors égal à $3{,}6.10^{-25}$/h. Même en ne considérant qu'un seul message, on voit que ce taux est largement inférieur au taux d'événement catastrophique qui est fixé à 10^{-9}/h.

Le risque d'avoir plusieurs messages erronés non détectés sur 10 messages consécutifs sera extrêmement faible dans la mesure où le bruit est par nature non répétitif dans le temps, et donc d'un message à l'autre. La probabilité d'un second message erroné et non détecté ne dépend pas du fait que l'on ait déjà eu une non détection du premier message erroné. Ainsi, le risque est de $3{,}6\ 10^{-25}.9.10^{-32} = 3{,}3.10^{-56}$ /h d'avoir dans un lot de 10 messages, 2 messages erronés et non détectés. Pour 3 messages erronés et non détectés, le taux est encore plus faible.

2.4.3 Risques associés aux défaillances du câblage

En considérant le modèle d'erreurs multiples indépendantes comme modèle d'erreur associé, cela revient à considérer un taux ε égal à 0,5 dans la formule précédente. Quand n est grand, la probabilité de non détection d'un message erroné ne dépend alors que de la longueur p du CRC et vaut 2^{-p}, soit 2^{-16} pour un CRC 16 bits. En considérant un taux de défaillance du câblage de 10^{-5}/h, cela conduit à un taux de messages erronés non détectés égal à $2^{-16}.10^{-5}$/h soit $1{,}5.10^{-10}$/h.

Même si ce taux n'est pas très éloigné d'un taux de 10^{-9}/h, c'est le fait de considérer X messages erronés non détectés sur 10 messages consécutifs comme événement redouté qui va permettre d'être très inférieur à ce taux. Comme il y a une indépendance entre les erreurs affectant des messages différents, le risque d'avoir X messages erronés non détectés sur 10 messages consécutifs est donné par :

$$1{,}5.10^{-10}.\prod_{i=1}^{X-1}(10-i).2^{-16}$$

Cela conduit à un taux de 2.10^{-14}/h pour $X = 2$, et de $2{,}4.10^{-18}$/h pour $X = 3$.

Cette première évaluation de la couverture des risques montre que les mécanismes de détection du CRC permettent d'aboutir à un taux très inférieur au seuil de criticité fixé pour les risques liés au bruit intrinsèque des composants ainsi qu'aux défaillances matérielles du câblage [Youssef *et al.* 2003b].

2.4.4 Risques associés aux défaillances des nœuds intermédiaires

Les risques associés à des nœuds intermédiaires *Ri* sont liés au type de fonction qu'ils assurent, fonction qui peut aller de la simple amplification du signal à des manipulations sur les messages reçus (mémorisation et retransmission des messages, ou aussi manipulation du contenu des messages). Dans le cas le plus simple, c'est-à-dire celui d'un nœud intermédiaire *Ri* assurant seulement une fonction de régénération du signal (simple répéteur), les risques sont alors, et comme illustré à la figure 2.6, les mêmes que ceux qui sont liés au câblage.

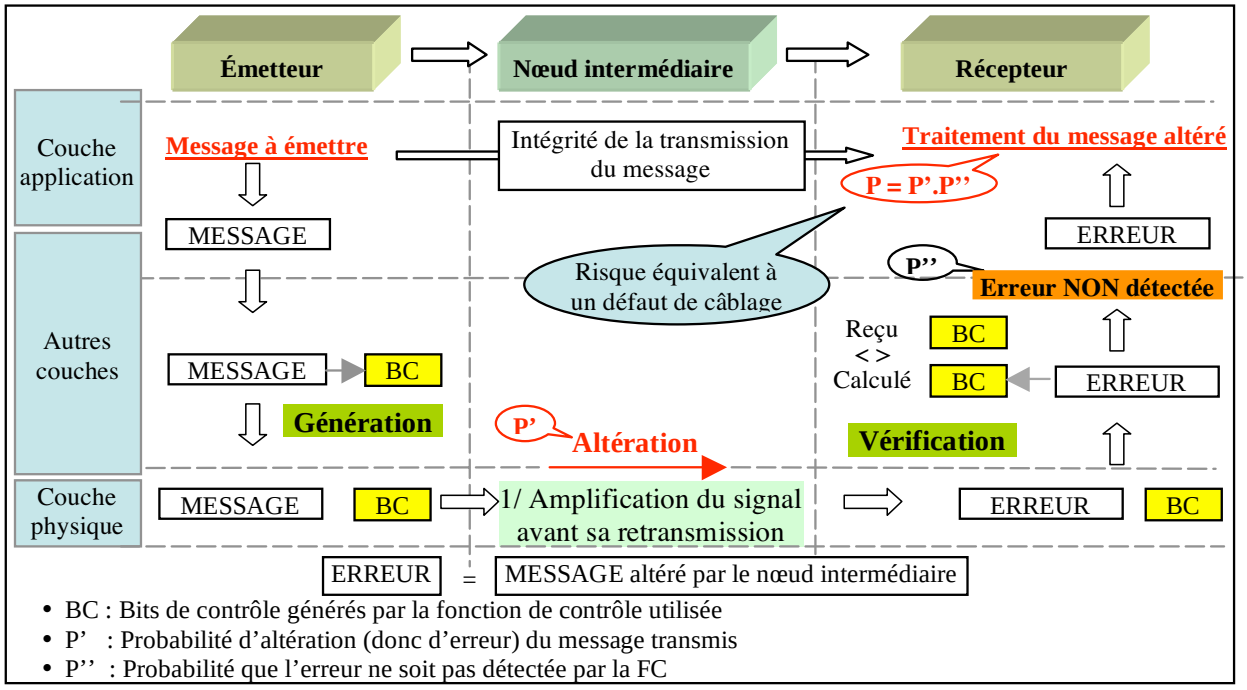


Figure 2.6 - Nœud intermédiaire en tant que simple répéteur

Lorsque la fonction du nœud intermédiaire nécessite de mémoriser le message avant sa retransmission, les risques sont plus importants. Ils sont liés à des fautes affectant la mémoire de stockage des messages (défaut physique ou d'accès mémoire en lecture ou en écriture).

2.4.4.1 Mémorisation des messages

Dans ce qui suit, nous distinguons deux types de fautes pouvant affecter la mémorisation des messages au niveau du nœud intermédiaire : les fautes du type "collage de bits" et les fautes d'adressage mémoire.

2.4.4.1.1 Collage de bits

La figure 2.7 illustre une faute *permanente* ou *temporaire* du tampon de stockage du nœud intermédiaire (qui entraîne l'altération du message mémorisé : collage de bits à 1 ou à 0), mais d'une durée suffisamment longue pour affecter plusieurs messages successifs.

Pour un seul message, les risques ne sont toutefois pas différents d'un défaut de câblage. Par contre, en raisonnant sur un lot de messages, et sachant que ce type d'erreur peut être répétitif d'un message à un autre, le risque alors d'avoir ***X*** messages erronés non détectés dans un lot de messages est important, et ce d'autant plus si les données ne changent pas d'un message à l'autre. En effet, dans ce dernier cas, la probabilité de ne pas détecter les *X-1* messages erronés après le premier est de 1, si l'altération affectant celui-ci n'a pas été détectée.

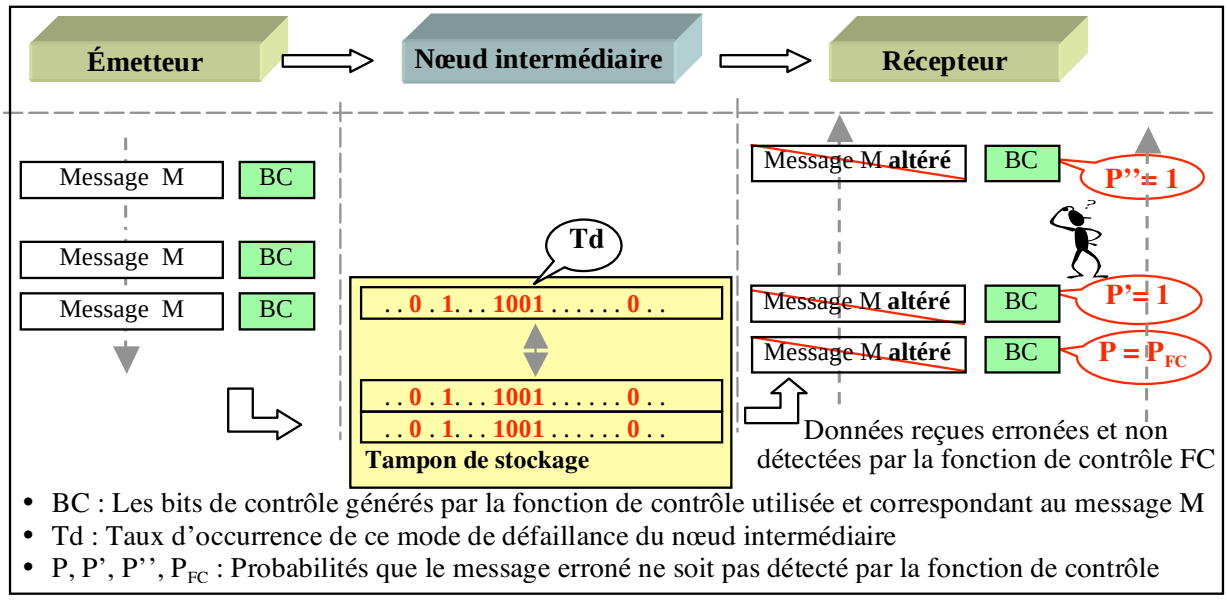


Figure 2.7 - Risques associés aux collages de bits

Afin d'analyser les risques du type "collage de bits" dans le cas des codes CRC, prenons un exemple concret. Supposons que le nœud émetteur transmette la séquence de k bits (k=10) de données suivante "1.1.0.1.0.1.1.0.1.1" et dont le polynôme associé $M(x)$ de degré $k-1$ est :

$$1.x^9+1.x^8+0.x^7+1.x^6+0.x^5+1.x^4+1.x^3+0.x^2+1.x+1 \;=\; x^9+x^8+x^6+x^4+x^3+x+1.$$

Enfin, pour générer les bits de contrôle, prenons le polynôme générateur $G(x) = x^4+x+1$. Ce qui donne, dans une représentation binaire :

- $M(x) = x^9+x^8+x^6+x^4+x^3+x+1 \;\Rightarrow$ 1.1.0.1.0.1.1.0.1.1
- $G(x) = x^4+x+1 \;\Rightarrow$ 1.0.0.1.1

L'exemple est illustré par la figure 2.8, qui décrit à la fois la génération des bits de CRC et le cas d'une erreur non détectée induite par une faute du type "collage de bits".

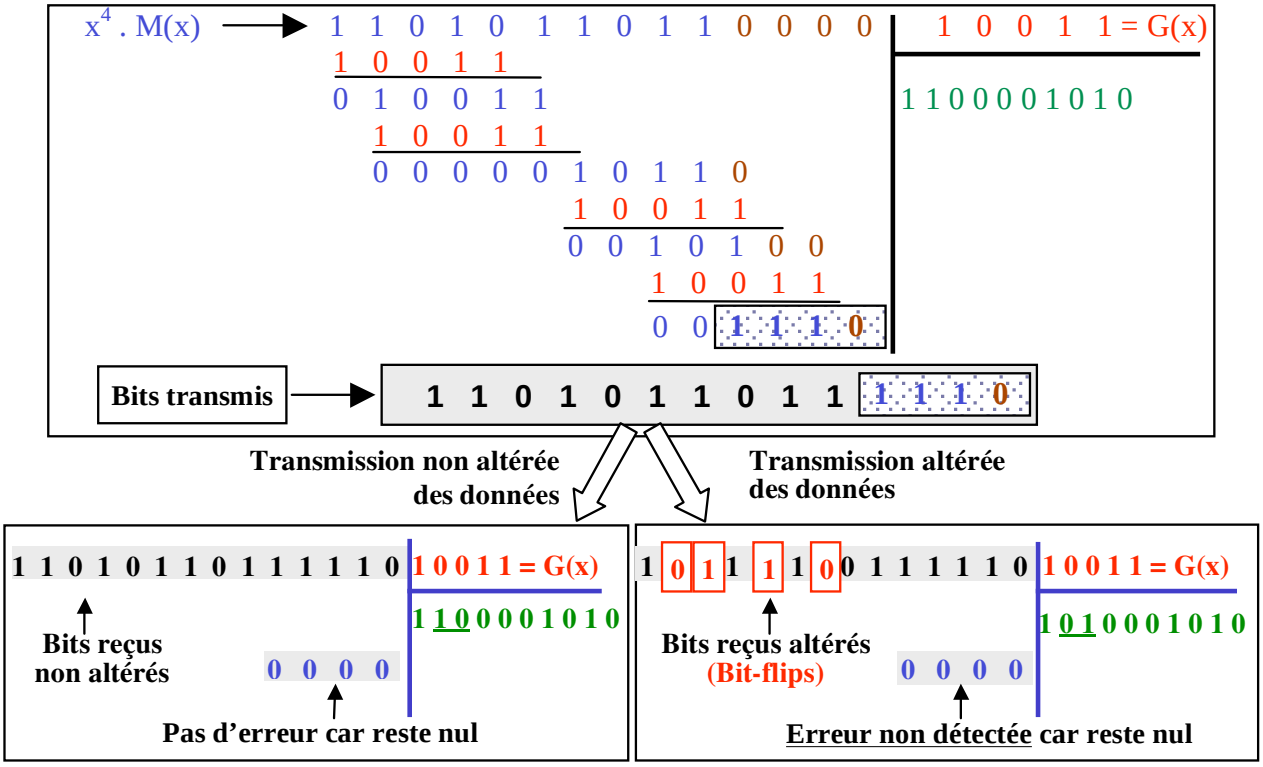


Figure 2.8 - Illustration d'un exemple d'erreur non détectée par le code CRC

En considérant l'erreur non détectée de la figure 2.8, supposons maintenant que le contenu des différents messages successifs ne varie pas. Dans ce cas, et en reprenant la figure 2.7, on aboutit au cas illustré par la figure 2.9.

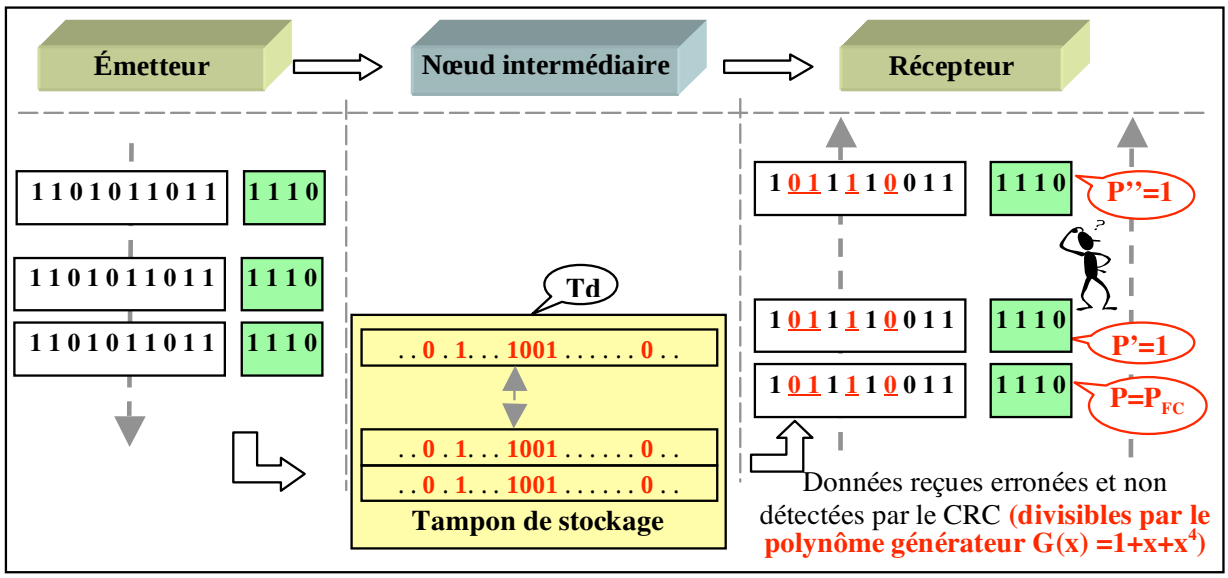


Figure 2.9 - Exemple des effets d'un collage répétitif de bits

En considérant un taux de défaillance des nœuds intermédiaires égal à 10^{-5}/h, et en cherchant à rester "conservatif", c'est-à-dire que toutes leurs défaillances se traduisent par des collages de bits au niveau de la mémoire, les résultats de l'évaluation de la couverture des risques liés à ces défaillances sont les suivants.

Dans le cas d'un message unitaire, on se retrouve dans la même situation que pour une défaillance du câblage. Avec un code CRC 16 bits, le taux de réception d'un message erroné non détecté est de $2^{-16}.10^{-5}$/h, soit $1{,}5.10^{-10}$/h.

En se plaçant au niveau d'un lot de messages, le pire cas consiste à considérer une erreur répétitive d'un message à un autre. Le risque alors d'avoir *X* messages erronés non détectés dans un lot de messages consécutifs ne diminue pas et reste égal à $1{,}5.10^{-10}$/h (la probabilité de ne pas détecter l'erreur sur les *X-1* messages suivants, sachant que l'erreur n'a pas été détectée sur le premier message, étant de 1).

Des valeurs moins conservatives, et sans doute plus proches de la réalité, pourraient être obtenues en multipliant le taux de défaillance d'un nœud par un facteur p_j qui correspond à la proportion des cas de défaillances qui se traduisent par un collage de bits au niveau de la mémoire. Ainsi, en prenant par exemple 1% pour le facteur p_j, on aboutit alors à un risque de $1{,}5.10^{-12}$/h.

2.4.4.1.2 Fautes d'adressage mémoire

Le contrôleur du nœud intermédiaire peut également présenter des risques vis-à-vis du rafraîchissement. Par exemple, si la mémoire de stockage n'est plus accessible en écriture, il peut toujours émettre le même message. Ainsi les données reçues par le nœud destinataire (surface de contrôle du système de CDV par exemple) ne présentent pas des erreurs en valeur, mais des erreurs temporelles, c'est-à-dire des données périmées ou non correctement rafraîchies), qui ne devraient donc pas être traitées par le nœud récepteur.

Pour ce type de défaillance du nœud intermédiaire, on pourrait considérer que toute défaillance peut se traduire par l'envoi de données périmées mais correctement codées, c'est-à-dire avec des bits de contrôle conformes avec les données retardées (cf. figure 2.10).

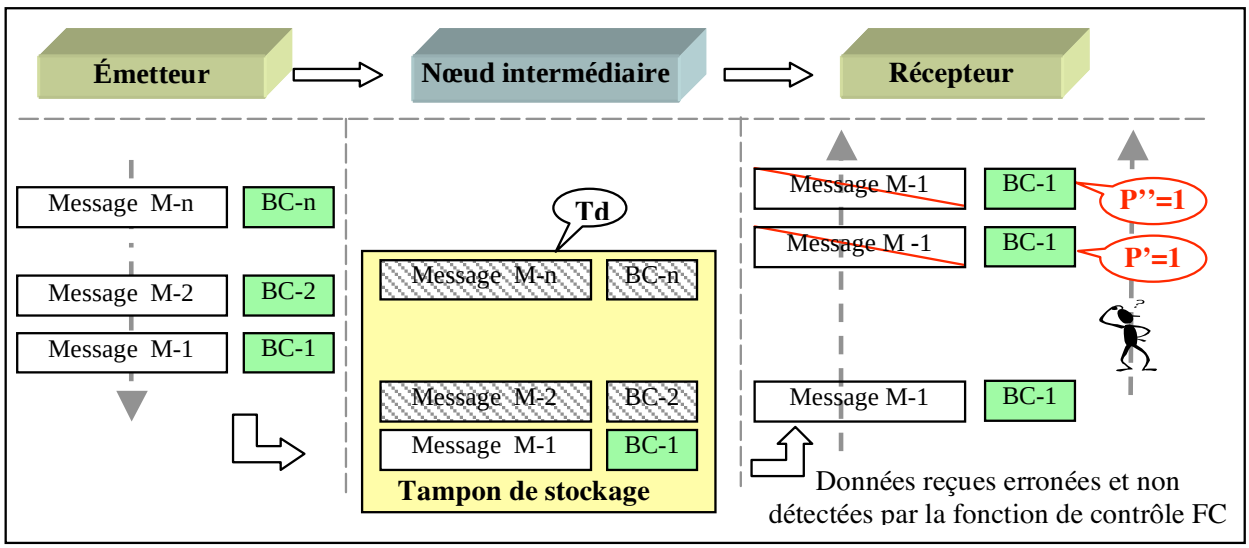


Figure 2.10 - Risques associés aux fautes d'adressage de la mémoire de stockage du nœud intermédiaire

La valeur du risque associé pour un ou plusieurs messages erronés est alors égal au taux de défaillance du nœud intermédiaire en considérant le pire cas où la défaillance du nœud conduit toujours au cas de fautes considéré ici. En prenant les mêmes valeurs que précédemment, on aboutit à une valeur du risque égale à 10^{-5}/h ou à 10^{-7}/h si seulement 1% des défaillances se traduit par le mode de défaillance concerné.

2.4.4.2 Manipulation du contenu du message

Dans le cas d'un nœud intermédiaire extrêmement complexe tel qu'un routeur, il peut être amené à interpréter et à manipuler certains champs du message. Les champs concernés sont des champs rajoutés par les couches qui se trouvent en dessous de la couche application et qui encapsulent la donnée du niveau application.

Dans ce cas, le nœud intermédiaire, après vérification du message reçu, modifie son contenu (par exemple, en ajustant les données réseau), recalcule le CRC et retransmet le message. Le nouveau CRC ne couvrira pas alors les erreurs éventuellement intervenues lors de la manipulation du message par le nœud intermédiaire (cf. figure 2.11).

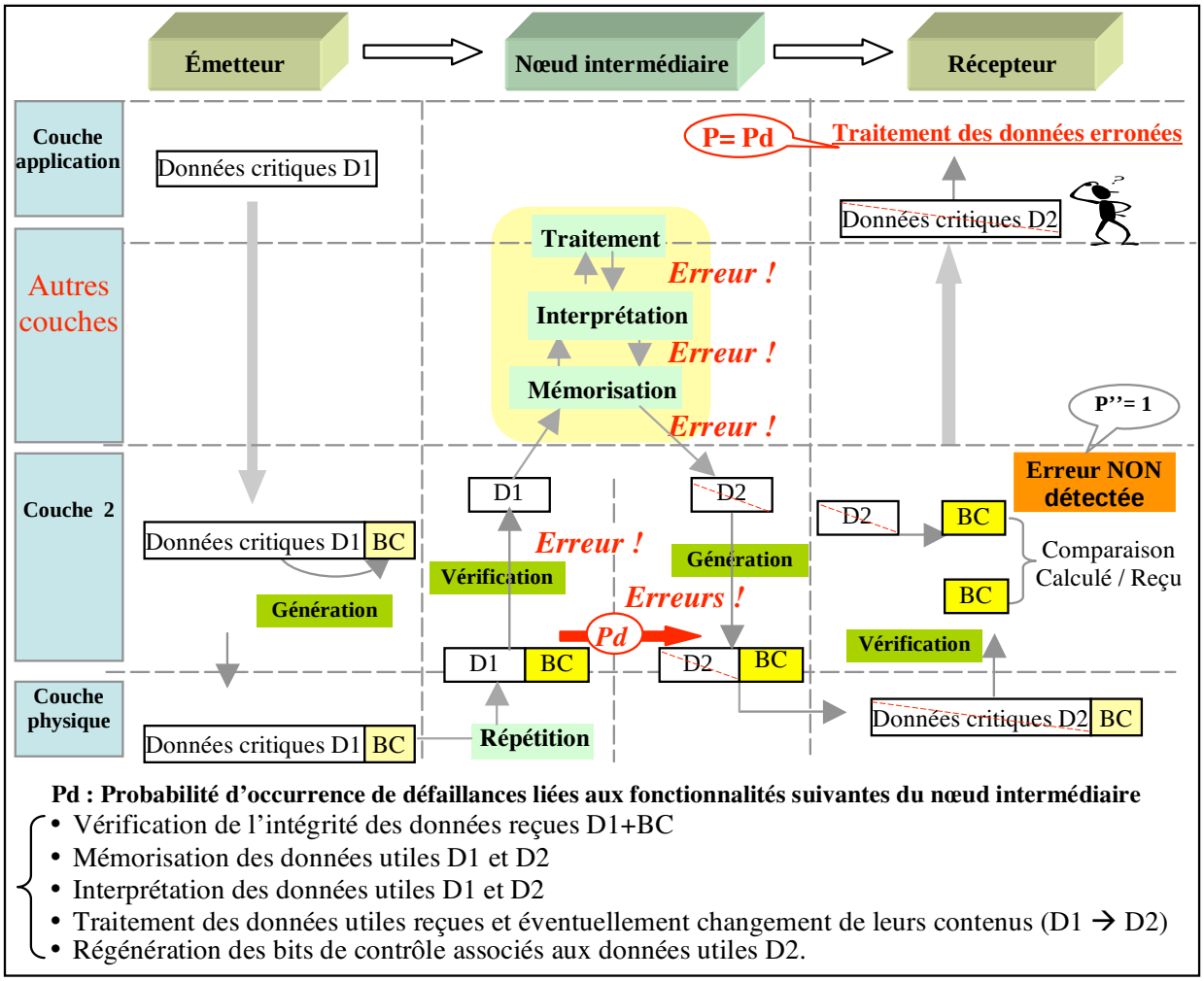


Figure 2.11 - Traitement du contenu du champ du message reçu par le nœud intermédiaire et risques associés

Concernant les risques associés, on pourrait considérer de manière conservative, que toute défaillance peut se traduire par l'envoi de données erronées avec un CRC conforme avec ces données. La valeur du risque associé pour un ou plusieurs messages erronés est alors égal au taux de défaillance du nœud intermédiaire. En prenant les mêmes valeurs que précédemment, on aboutit à une valeur du risque égale à 10^{-5}/h ou à 10^{-7}/h si seulement 1% des défaillances se traduit par le mode de défaillance concerné.

Cette fois également la fonction de contrôle utilisée pour détecter les éventuelles altérations des données transitant par le nœud intermédiaire ne couvre plus les risques considérés. Il est également important de noter, à ce stade de l'analyse, que l'intégrité des données échangées de bout en bout du système de communication ne peut pas être assurée si le nœud intermédiaire a la possibilité de générer des bits de contrôle relatifs à des messages qu'il a reçus, mémorisés, interprétés et manipulés (cf. figure 2.11). Dans ce cas, certaines défaillances du nœud intermédiaire peuvent se traduire par la transmission de données altérées, mais correctement codées (les bits de contrôle étant générés sur la base de données erronées) et le nœud récepteur n'a donc aucun moyen de détecter ces altérations.

2.4.5 Bilan des évaluations

Les évaluations qui ont été effectuées, concernant les différents types d'altérations, montrent le fort impact du modèle d'erreur associé à un type d'altérations sur le risque associé.

Pour des altérations liées au bruit (erreurs aléatoires indépendantes) ou aux défauts de câblage (erreurs multiples indépendantes), même si les risques au niveau d'un message unitaire pouvaient être à la limite de l'acceptable, c'est le fait de considérer le risque au niveau d'un lot de messages et le fait que l'altération ne soit pas répétitive qui fait que le risque est nettement inférieur au seuil fixé.

Dans le cas des défaillances fonctionnelles des nœuds intermédiaires, ce n'est plus vrai compte tenu de la possible répétitivité de l'altération (erreurs multiples répétitives), qui fait que le risque associé à l'altération d'un lot de messages est le même que celui associé à l'altération d'un message unitaire. Si au niveau d'un nœud intermédiaire, on ne considère que des fonctions de mémorisation, les codes CRC apportent une certaine protection qui pourrait être suffisante suivant le nombre de bits de CRC ; par contre, si le nœud intermédiaire présente des fonctionnalités de traitement réseau, ce n'est plus le cas.

2.5 Conclusion

Ce chapitre nous a tout d'abord permis de présenter l'architecture de communication envisagée dans le cadre de systèmes de commande-contrôle avec des nappes de microsystèmes. Il s'agit d'une architecture hiérarchique, avec nécessairement des nœuds intermédiaires entre les calculateurs de commande et les nœuds commandés. Au niveau du type de protocole envisagé, la criticité des systèmes visés rend nécessaire l'utilisation de protocoles synchrones.

L'identification des risques que nous avons ensuite effectuée nous a permis d'établir différents modèles d'erreur en fonction du type de fautes considéré. L'évaluation des risques qui a suivi a montré que, en présence de nœuds intermédiaires, la protection standard par codes CRC ne permet pas de garantir le niveau d'intégrité visé sur une voie de communication. D'une part, cela est lié au type d'erreurs engendrées par des fautes du nœud intermédiaire (erreurs multiples répétitives). Cela est lié aussi au fait que, d'autre part, dans le cas de nœuds complexes ou actifs, les bits de contrôle pouvant être recalculés localement, les erreurs intervenant lors des opérations de traitement réseau ne sont pas couvertes. Le taux de l'événement redouté est, dans ce cas, égal au taux de défaillance du nœud intermédiaire.

Ces résultats sont en total accord avec une étude publiée récemment [Paulitsch *et al*. 2005] qui montre également les problèmes liés aux nœuds intermédiaires. Leurs recommandations vont jusqu'à ne considérer l'utilisation de codes CRC que pour la protection vis-à-vis des erreurs du bus de communication (erreurs liées au bruit ou aux perturbations électromagnétiques).

En conséquence, pour atteindre le niveau d'intégrité visé pour le système de communication, il nous faut donc proposer d'autres techniques de protection pour couvrir, en particulier, les risques liés aux défaillances des nœuds intermédiaires qui ne sont pas suffisamment couverts, ou pas du tout, par les codes CRC mis en œuvre au niveau des couches basses d'un réseau de communication. Cela fait l'objet du chapitre suivant.

Chapitre 3 - Fonction de contrôle d'erreur évolutive pour un haut niveau d'intégrité

Ce chapitre vise à proposer des techniques, mises en œuvre au niveau de la couche application des nœuds émetteurs et récepteurs d'un système de communication, pour assurer un haut niveau d'intégrité des données applicatives échangées. Ces techniques font abstraction des couches basses d'un système de communication et des duplications éventuelles d'un ou plusieurs de ses composants (bus de communication, composant d'un nœud, nœud ou groupes de nœuds). En effet, notre but est de proposer, puis de valider une solution logicielle qui permette de satisfaire les différents aspects d'une communication intègre, dont l'étude des risques réalisée au chapitre précédent a montré qu'ils ne sont pas couverts, notamment en présence de certains modes de défaillances fonctionnelles des nœuds intermédiaires.

Le choix d'une solution purement logicielle est lié à nos besoins de garantir, d'une part, une flexibilité de mise en œuvre, et d'autre part d'assurer un haut niveau d'intégrité des communications entre les couches applications des nœuds émetteurs et récepteurs, ou entre les nœuds de commande (calculateurs) et les nœuds commandés (actionneurs) dans le cas des systèmes de commande-contrôle. Cette mise en œuvre logicielle permet également de se prémunir contre d'éventuels problèmes de conception matérielle, et de ne pas être dépendant des solutions spécifiques utilisées dans les couches physiques.

Dans ce chapitre, nous proposons des techniques de contrôle des erreurs en valeur, d'adressage et temporelles. La plus grande partie de ce chapitre a pour but la spécification d'une fonction de contrôle d'erreur qui assure un grand pouvoir de détection d'erreurs, et ceci en prenant en compte l'intégrité par rapport à un lot de messages et non par rapport à un message seul. En effet, dans les types d'applications visées, un scénario de défaillance redouté suppose qu'une altération non détectée soit appliquée trop longtemps et donc sur un lot de messages.

La dernière partie de ce chapitre présente l'application des techniques proposées au système de communication pour les commandes de vol, dans le but de les valider par rapport aux objectifs d'intégrité fixés.

3.1 Fonction de contrôle d'erreur

Les messages transmis de bout en bout du système de communication peuvent être affectés par différents types d'erreur. Parmi ces erreurs, il y a celles qui sont aléatoires, dues au bruit du canal de transmission ou au bruit des composants, et celles qui sont en rafales, dues à une rupture ou à un court-circuit d'un connecteur ou d'un câble. Mais, il y a aussi les erreurs qui peuvent résulter d'une défaillance d'un nœud intermédiaire et qui, comme nous avons vu au chapitre 2, posent des problèmes compte tenu du fait qu'elles peuvent se reproduire sur plusieurs messages successifs en raison de leur caractère permanent.

Le but de cette section est de présenter les fonctions de contrôle d'erreur envisagées au niveau de la couche applicative, et permettant de faire face aux défaillances des nœuds intermédiaires et aux insuffisances des techniques de protection (détection d'erreurs par code CRC) mises en œuvre au niveau des couches matérielles. En se référant aux spécifications des propriétés d'une communication intègre pour les SCC considérés (cf. § 1.4.1.3), il nous faut proposer une fonction de contrôle permettant de traiter les erreurs **en valeur**, d'**adressage** et **temporelles**.

Dans un premier temps, nous considèrerons que les nœuds intermédiaires n'ont aucun rôle au niveau applicatif, et n'ont donc pas à connaître les fonctions de contrôle d'erreur utilisées au niveau applicatif. En fin de chapitre, nous présentons le cas où ces nœuds ont un rôle applicatif.

Mettre en place des mécanismes de contrôle d'erreur de bout en bout au niveau applicatif est une pratique courante. Cela permet de faire abstraction des couches inférieures d'un système, et donc de ne pas s'appuyer sur les techniques de détection d'erreurs des niveaux inférieurs. Par exemple, citons le mode HEDC *(High Error Detection Coverage)* proposé pour des systèmes basés sur le protocole TTP [Kopetz 1998], ou la couche de sécurité (*Safety Layer*) proposée dans [Brodtkorb 2001] pour le bus de terrain de la *Foundation Fieldbus*.

Nous pouvons aussi citer le mécanisme mis en œuvre dans le cadre de l'architecture GUARDS. Ce mécanisme, appelé "*Keyed CRC"* [Powell 2001, pp. 45-50], permet de faire l'hypothèse que, si un nœud B reçoit un message d'un nœud A, il ne peut communiquer une version altérée du message à un nœud C sans que celui-ci soit incapable de détecter cette altération. Pour cela, le nœud A code le message avec une clé qu'il partage avec le nœud C, mais que le nœud B ne connaît pas. Dans notre cas, les nœuds A et C correspondent respectivement aux nœuds émetteur et récepteur, et le nœud B correspond au nœud intermédiaire.

Notre proposition d'utiliser une fonction de contrôle de bout en bout n'est donc pas une contribution majeure. Mais sur ce plan, nous fournissons ci-après des indications permettant de tenir compte au sein d'une unique fonction de contrôle des divers types d'erreur considérés.

Notre véritable contribution se situe au niveau de la proposition faite au § 3.3 pour aboutir à un haut niveau d'intégrité avec un coût faible en tirant profit du fait que, dans notre cas, l'intégrité est considérée sur un lot de messages et non sur un message unitaire.

3.1.1 Contrôle des erreurs en valeur

La technique présentée dans cette section a pour but d'assurer un pouvoir de détection des erreurs en valeur. Elle se base sur le principe des fonctions de hachage, qui génèrent un condensé du message (haché), obtenu par calcul à partir du message reçu. L'association "message - condensé" est conçue de telle sorte qu'un petit changement du message produise un gros changement de son haché [Bellare *et al.* 1996a]. Comme exemples de fonctions de hachage, on peut citer celles qui génèrent les codes CRC.

En effet, en se basant sur des polynômes générateurs de degré q, ces fonctions génèrent des mots condensés ou codes CRC de longueur q à partir d'un message en entrée qui peut être de longueur variable. L'utilisation des algorithmes de génération logicielle de ces codes est présentée dans le chapitre 4.

Le processus de contrôle des erreurs en valeur est le suivant :

- le nœud émetteur utilise la fonction de hachage pour calculer le haché relatif à la partie du message à émettre et dont il veut assurer un contrôle de ses éventuelles altérations,
- le nœud récepteur recalcule le haché en utilisant le même procédé que le nœud émetteur et le compare à celui qui accompagne le message reçu. Si les hachés calculé et reçu sont différents, alors le nœud récepteur rejette le message.

3.1.2 Contrôle des erreurs d'adressage

Tout d'abord, il faut préciser qu'avec la technique de contrôle des erreurs d'adressage présentée ci-dessous, nous ne cherchons pas une protection des erreurs intentionnelles dues par exemple à l'usurpation d'identité par un intrus se connectant au système de communication. Nous cherchons plutôt à proposer une technique d'identification des messages, qui permette pour chaque nœud du système de communication de vérifier le lien entre le contenu d'un message reçu, son origine et sa destination. Le but étant de permettre à tout nœud récepteur d'un message de vérifier s'il provient d'un émetteur censé communiquer avec lui, et surtout vérifier que le message reçu lui est bien destiné. Dans le cadre de notre cas d'étude, nous considérerons qu'il n'y a qu'un seul nœud émetteur (le calculateur des CDV), mais que celui-ci peut envoyer des messages à plusieurs nœuds récepteurs (organes de commande des surfaces).

La technique de contrôle des erreurs d'adressage retenue se base sur le principe de la fonction de hachage, et sur le fait qu'un *code* (ou clé) est associé à tout échange de messages entre deux nœuds du système de communication [Bellare *et al.* 1996b].

Pour identifier des messages échangés dans le cadre d'un type particulier de communication, le code utilisé doit être connu par tous les nœuds impliqués. Il y aura donc autant de codes que de types de communications. Si par contre, tout échange de données entre deux nœuds représente un type particulier de communication, il y aura alors autant de codes que de combinaisons de communications point à point possibles.

L'avantage de cette technique est qu'elle ne nécessite pas la transmission du code, vu qu'il est connu à l'avance par tous les émetteurs et les récepteurs. Un exemple de code peut être construit par la concaténation des suites de bits caractérisant chaque type de communication.

3.1.3 Contrôle des erreurs temporelles

La technique de contrôle des erreurs temporelles présentée ci-dessous permet d'assurer une vérification systématique de la non péremption temporelle pour chaque message reçu. Les objectifs étant, d'une part, de vérifier que la latence de transmission d'un message de bout en bout du système de communication répond aux objectifs temps réel fixés et, d'autre part, de pouvoir traiter la défaillance fonctionnelle d'un noeud émettant des données périmées.

En raisonnant sur un SCC, ces deux objectifs garantissent un temps de réponse vis-à-vis des informations reçues de l'environnement proche du système, et véhiculées par l'ensemble des capteurs qui peuvent lui être indirectement reliés. Et dans un temps de réponse qui satisfait les contraintes temps réel des applications qu'il supporte.

La technique de contrôle des erreurs temporelles se base sur le principe de datation des messages et plus précisément sur la notion du temps global qui s'applique aux systèmes distribués. Dans ces systèmes, deux politiques peuvent être adoptées vis-à-vis du temps. La première, représentant le temps local, se base sur le fait que chaque nœud ou groupe de nœuds (sous système ou site) possède sa propre référence temporelle et ne connaît pas celle des autres. La seconde, représentant le temps global, impose une référence temporelle commune à tous les nœuds. Il y a deux solutions pour fournir ce temps global : soit par une horloge centralisée, soit par la synchronisation des temps locaux des différents nœuds ou sites.

La première solution est généralement rejetée car elle n'est pas satisfaisante du point de vue sûreté de fonctionnement, vu que l'horloge centralisée constitue un point de défaillance dur du système. La plupart des protocoles de communication (CAN, TTP/C, …) optent donc pour la seconde solution et fournissement des mécanismes de synchronisation matérielle des temps locaux, basées sur les signaux d'horloges [Smith 1981].

Dans le cadre de notre étude, nous cherchons à proposer une technique au niveau de la couche application pour faire face les éventuelles défaillances de ces mécanismes, dues notamment à des problèmes de synchronisation d'horloges [Pfeifer *et al.* 1999, Sivencrona *et al.* 2000].

La technique de contrôle des erreurs temporelles se base sur la fonction de hachage décrite précédemment. L'idée est d'utiliser un compteur qui s'initialise à chaque phase de synchronisation et qui s'incrémente de façon périodique. La période d'incrémentation doit être inférieur au délai maximum toléré avant de considérer les données reçues comme périmées. Pour les systèmes de CDV, où les surfaces de contrôle sont à rafraîchir tous les 10 ms, la période d'incrémentation du compteur doit être donc strictement inférieure au seuil des 10 ms.

Comme pour la technique de contrôle des erreurs d'adressage, le champ *compteur* ou *indice temporel* est utilisé pour calculer le haché, mais sa transmission est inutile vu que sa valeur est identique du côté de l'émetteur et du récepteur (supposés être correctement synchronisés).

3.1.4 Fonction globale de contrôle

Les trois types de contrôle d'erreur qui viennent d'être présentés, et qui reposent sur le principe de la fonction de hachage, peuvent être combinés pour former une fonction globale de contrôle assurant une protection contre les erreurs en valeur, d'adressage et temporelles.

Dans ce cas, le calcul de haché côté émetteur et sa vérification côté récepteur se basent sur la concaténation de trois champs (cf. figure 3.1) : le champ des données émises dont on cherche à assurer l'intégrité (généralement des données critiques ou utiles), le champ *adresse* (ou code) et le champ *indice temporel*.

Notons que dans le cadre de notre étude basée sur l'intégrité des communications, la fonction de hachage représente une fonction de contrôle d'erreur, et le haché représente le code généré par cette fonction permettant de détecter les éventuelles erreurs de transmission.

Quelle que soit la fonction de hachage utilisée, la probabilité de non détection d'un message erroné, en présence d'erreurs multiples répétitives liées aux défaillances fonctionnelles des nœuds intermédiaires (cf. § 2.4.4), dépend de la taille du haché ; elle est égale à $\mathbf{2^{-p}}$, où p correspond au nombre de bits du haché (ou bits de contrôle). Avec p égal à 16, ce qui nous semble un bon compromis au niveau rendement d'un code dans le cadre de notre cas d'étude, une évaluation des risques équivalente à l'évaluation faite au chapitre 2 montre que le pouvoir de détection de la fonction globale est limite pour garantir les objectifs d'intégrité visés.

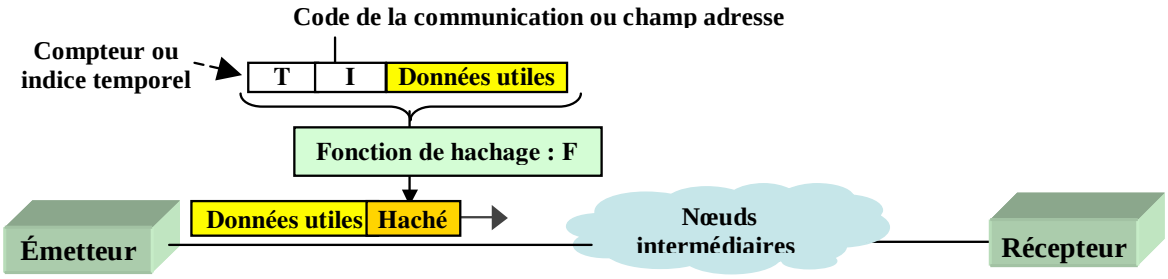


Figure 3.1 - Fonction globale de contrôle d'erreur

Une partie importante du travail a donc porté sur la définition, puis la validation des techniques logicielles complémentaires pour augmenter le pouvoir de détection d'erreurs et assurer ainsi un plus haut niveau d'intégrité, sachant que dans notre démarche, on évite de faire appel à des redondances matérielles (par ex. bus doublés) ou à des duplications des messages. Nous allons d'abord analyser les solutions classiques avant de présenter la solution que nous proposons.

3.2 Solutions classiques d'augmentation du pouvoir détection

L'augmentation du pouvoir de détection d'un code détecteur d'erreurs passe inévitablement par l'augmentation de la redondance utilisée, et donc du nombre de bits de contrôle. Cela peut être mis en œuvre par une simple augmentation du nombre des bits de contrôle, ou en utilisant la technique de fragmentation. Nous présentons successivement ces deux possibilités.

3.2.1 Simple augmentation du nombre des bits de contrôle

Pour montrer l'apport de l'augmentation du nombre des bits de contrôle sur le pouvoir de détection d'erreurs, nous prenons le cas particulier des erreurs aléatoires. Nous rappelons ci-dessous l'expression déjà vue au chapitre 2, et qui est valable quel que soit le code détecteur en bloc utilisé. Elle donne, pour un code de longueur n, la probabilité P_{ue} de non détection des messages erronés sur un canal binaire symétrique, avec un taux d'erreur bit, noté ε ($0 <= \varepsilon <= 0.5$), pouvant générer des erreurs aléatoires [Castagnoli *et al.* 1990].

$$P_{ue}(\varepsilon, n) = \sum_{i=d}^{n} (A_i * (\varepsilon^i) * (1-\varepsilon)^{(n-i)}) \quad \sim= A_d * \varepsilon^d \quad (\text{pour } \varepsilon << 1)$$

$$\sim= 2^{-p} - 2^{-n} \quad (\text{pour } \varepsilon = 0{,}5)$$

avec : A_i le nombre de mots de code de poids i (c'est-à-dire avec i bits à 1), d la distance de Hamming du code, p le nombre de bits de contrôle.

À partir de cette formule, on remarque que le pouvoir de détection d'erreurs aléatoires d'un code C augmente avec la distance de Hamming, et que cette augmentation est d'autant plus importante que la valeur de ε est faible. Généralement, un code présentant une distance de Hamming de 4 est jugé bon pour la détection des erreurs aléatoires indépendantes [Stark 2000].

Pour analyser comment évolue le pouvoir de détection avec le nombre p de bits de contrôle, il faut d'abord examiner pour différentes valeurs de p, quelle est la distance de Hamming maximum qu'il est possible d'obtenir en fonction de la longueur n du code. C'est ce que montre la figure 3.2, établie à partir de plusieurs études comparatives [Merley & Posner 1984, Castagnoli *et al.* 1990, Boudreau *et al.* 1994] entre plusieurs codes CRC, académiques et industriels, pour des codes CRC 16, 24 et 32 bits.

Par exemple, pour les codes CRC 16 bits, la distance de Hamming est de 6 pour une longueur n comprise entre 36 et 151, et elle est de 4 pour une longueur *n* comprise entre 258 et 2^{15}-1.

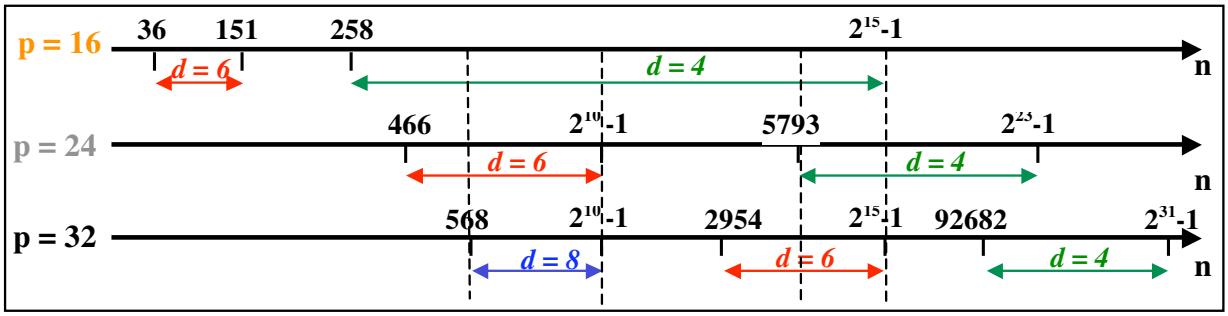


Figure 3.2 - Distance de Hamming et longueur n pour des codes CRC 16, 24 et 32 bits

Représentons différemment ces résultats pour montrer l'évolution de la distance de Hamming d'un code CRC suivant le nombre de bits de contrôle (cf. figure 3.3) pour deux intervalles différents de *n* matérialisés sur la figure 3.2 par des lignes verticales en pointillé.

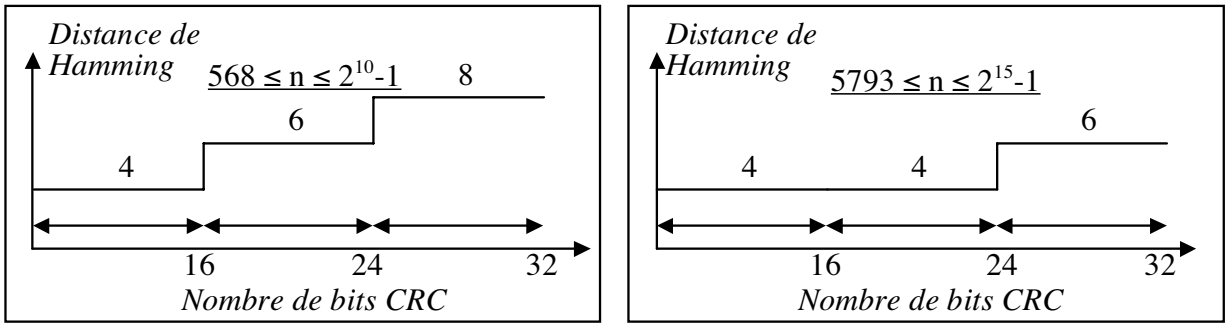


Figure 3.3 - Distance de Hamming (nombre de bits de CRC)

La figure 3.3 montre que, pour une longueur *n* des messages codés comprise entre 568 et 2^{10}-1, l'augmentation du nombre des bits de contrôle d'un code CRC de 16 à 24, puis de 24 à 32, augmente respectivement sa distance de Hamming de 4 à 6 et de 6 à 8. Alors, et vu que nous avons fait l'hypothèse que la valeur de ε est faible, les pouvoirs de détection d'erreurs aléatoires augmentent de façon importante en passant successivement de 16 bits de CRC à 24 bits, puis 32 bits. Pour une longueur *n* comprise entre 5793 et 2^{15}-1, la distance de Hamming ne change pas entre 16 et 24 bits de CRC.

Mais pour les études résumées précédemment, *n* est toujours supérieur à 568. Ces études ne permettent donc pas d'avoir des indications pour notre cas d'étude, pour lequel nous avons pris un nombre de bits utiles égal à 100, ce qui correspond respectivement à une longueur *n* de 116, 124 et 132 pour des codes CRC à 16, 24 et 32 bits. Cependant, nous avons trouvé les valeurs de *n* qui nous faisaient défaut dans une autre étude sur des codes CRC 24 et 32 bits [Castagnoli *et al.* 1993], portant sur 7 codes académiques dont 2 codes issus de [Merley & Posner 1984] et le code standard 32 bits IEEE 802. Nous n'avons pas inclus les valeurs de cette étude sur les figures précédentes en raison de la dispersion des valeurs de *n* suivant le polynôme générateur utilisé. Nous retenons simplement que, pour tous les codes CRC 24 bits et pour *n* = 124, la distance de Hamming est de 6, et reste la même que pour des codes CRC 16 bits. Et pour les codes CRC 32 bits et pour *n* = 132, la distance de Hamming varie entre 6 et 8 suivant le polynôme choisi, ce qui montre le fort impact que peut avoir le polynôme retenu.

Regardons maintenant l'impact de l'augmentation du nombre de bits de contrôle sur la probabilité de non détection dans le cas de notre cas d'étude.

Pour les erreurs aléatoires indépendantes ($\varepsilon << 1$), en appliquant la formule présentée précédemment, nous pouvons constater que la probabilité de non détection d'erreurs augmente lorsque l'on passe d'un CRC 16 bits à un CRC 24 bits : elle est respectivement de $2{,}97.10^{-39}$ et $4{,}47.10^{-39}$ pour $\varepsilon = 10^{-8}$ qui correspond à la valeur retenue pour notre cas d'étude. Ce résultat est pour le moins surprenant. Mais une réponse logique à cette contradiction est le fait que la formule donne un résultat pessimiste en se basant uniquement sur la distance de Hamming du code qui, rappelons le, correspond à la distance minimum entre **tous** les mots du code. Ainsi, s'il n'y a juste que quelques mots qui ne permettent pas d'afficher une distance de Hamming supérieure, le résultat sera forcément pessimiste. Ainsi, avec d = 7, la formule donne une valeur de $P_{ue} = 7{,}53.10^{-46}$. En ce qui concerne le passage à un code CRC 32 bits, le constat est le même. La valeur de P_{ue} est respectivement de $6{,}55.10^{-39}$, $1{,}18.10^{-45}$ et $1{,}84.10^{-52}$ pour une distance de Hamming respectivement de 6, 7 ou 8. Même si les écarts entre les différentes valeurs de la probabilité de non détection sont importants, nous n'avons pas besoin heureusement de nous en soucier car, même avec la valeur la plus pessimiste, le risque associé aux erreurs aléatoires indépendantes est largement couvert.

Pour les erreurs en rafale (*burst*), la taille du *burst* que le code permet de détecter à 100% peut ne pas évoluer en passant de 16 à 24 ou 32 bits, puisqu'elle est directement liée à la distance de Hamming du code. Dans [RFC 3385], on trouve une formule donnant la probabilité de non détection d'un *burst* en fonction de sa longueur. Nous n'avons pas cherché à l'utiliser sachant par ailleurs que, les erreurs en rafale n'entraînant pas un effet répétitif d'un message à un autre, elles ne posent pas non plus de problème particulier dans le cadre de notre cas d'étude.

Pour les erreurs multiples, qui consiste à considérer ε = 0,5, l'application de la formule conduit respectivement à des valeurs de P_{ue} de $1{,}52.10^{-5}$, $5{,}90.10^{-8}$ et $2{,}38.10^{-10}$ pour des codes CRC 16, 24 et 32 bits.

3.2.2 Fragmentation

La fragmentation représente une autre solution pour augmenter le nombre de bits de contrôle associés à un message. Elle consiste à calculer plusieurs champs de bits de contrôle sur un même message, chaque champ étant relatif à un fragment du message à émettre.

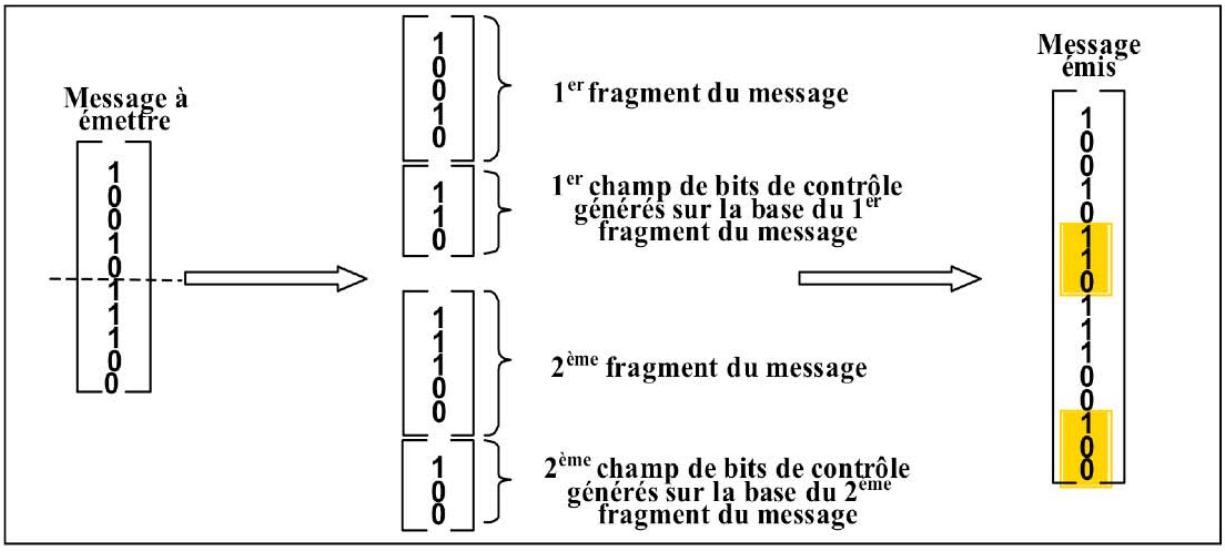


Figure 3.4 - Génération par fragmentation du mot codé à l'émission

Prenons l'exemple de messages ayant une taille initiale (avant codage) de 10 bits. Le message initial est, par exemple, découpé en deux fragments auxquels sont ajoutés un champ de 3 bits de contrôle. La taille du message codé émis est donc de 5 + 3 + 5 + 3 = 16 bits. À l'émission, la génération de ce message s'effectue selon le principe présenté à la figure 3.4.

À la réception, des bits de contrôle sur chaque fragment de message reçu sont calculés et comparés avec les bits de contrôle reçus (figure 3.5).

Le champ de contrôle total permet de vérifier les éventuelles altérations du message reçu. Il est sous la forme d'un vecteur ayant une taille égale au nombre de fragments par message. Une valeur de 1 du champ de contrôle total indique une erreur sur le champ de bits de contrôle associé à un fragment et une valeur de 0 indique que le champ de bits de contrôle reçu et celui calculé sont identiques et donc en principe, il n'y a pas eu d'altération lors de la transmission de ce fragment de message.

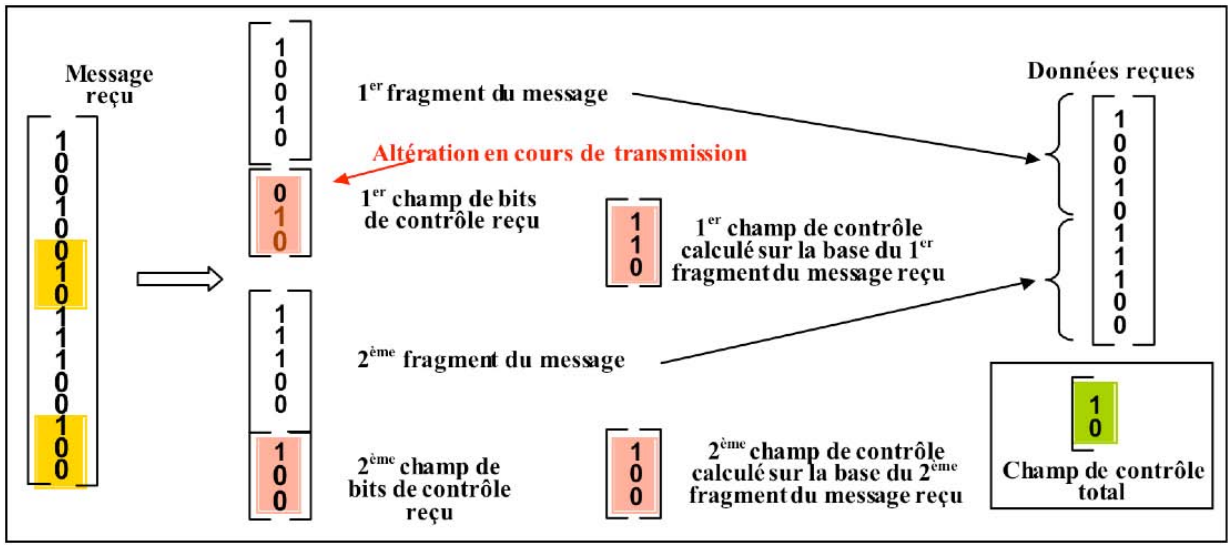


Figure 3.5 - Vérification du message codé à la réception

La technique de la fragmentation des messages est utilisée depuis longtemps dans Internet et apparaît dans la RFC qui décrit le standard du protocole Internet [RFC 791]. Dans le cas d'Internet, le principe de la fragmentation est surtout utilisé lorsqu'un message de grande taille arrive sur une portion de réseau qui n'accepte la transmission que de messages plus courts.

Dans ce qui suit, nous allons regarder quels avantages nous pourrions tirer de la fragmentation des bits de contrôle dans notre cas. Regardons d'abord l'impact sur le pouvoir de détection d'erreurs du code, et principalement dans le cas des erreurs multiples répétitives. Pour cela, reprenons l'exemple d'un canal binaire symétrique (avec donc un taux d'erreur bit ε de à 0,5), ainsi que la formule de la probabilité de non détection associée dans ce cas : $P_{ue} = 2^{-p} - 2^{-n}$. En se référant à l'exemple des figures 3.4 et 3.5 décrivant la technique de la fragmentation, pour un message de longueur 16, dont 6 bits sont alloués aux champx de contrôle, la probabilité de non détection d'erreurs correspond au produit des probabilités de non détection des erreurs dans les 2 fragments du message, c'est-à-dire $(2^{-3}-2^{-8})^2 = 0{,}0146$. Par contre, si on n'utilisait pas la technique de la fragmentation, la probabilité de non détection d'erreurs serait supérieure : $(2^{-6}-2^{-16}) = 0{,}0156$. A nombre identique de bits de contrôle, cela donne un léger avantage à la fragmentation dans le cas de cet exemple. Pour des valeurs de n plus importantes, l'écart est non significatif, même si théoriquement la fragmentation conduira à une valeur P_{ue} plus faible.

Par contre, la fragmentation présente deux avantages plus importants. Le premier est la possibilité de localiser une erreur au niveau d'un fragment de message permettant ainsi de ne pas rejeter le message dans sa totalité. Le deuxième est au niveau du temps de génération des bits de contrôle. En effet, visant une mise en œuvre logicielle au niveau applicatif, il est fort probable que le temps de génération des bits de contrôle ne croisse pas linéairement avec le nombre de bits de contrôle, et ce compte tenu de la taille plutôt limitée des registres au niveau des circuits microcontrôleurs utilisés. Ainsi, dans le cas d'un doublement du nombre de bits de contrôle (ex. : passage de 16 bits à 32), le temps de génération sera doublé dans le cas de l'utilisation de deux fragments, alors qu'il sera sans doute plus que doublé dans le cas d'une simple augmentation des bits de contrôle.

En conclusion, même si la fragmentation n'amène pas de gain significatif en termes de pouvoir de détection d'erreurs par rapport à la technique d'une simple augmentation des bits de contrôle, elle mérite quand même d'être considérée car elle présente quelques avantages. L'apport de la fragmentation est, par contre, plus important dans le cadre de la correction d'erreurs. En effet, dans le cas de canaux de communication très bruités, comme cela peut être le cas des communications sans fil, le taux d'erreur peut être tel que la correction des erreurs par détection et retransmission peut se révéler peu efficace, ceci en raison du nombre de retransmissions qui peuvent être nécessaires pour assurer le transfert d'un seul message [Denz & Nilsson 1998]. Le découpage de longs messages en plusieurs fragments facilite l'auto-correction puisque la multiplicité des erreurs sera plus faible sur un fragment, et seule la retransmission des fragments qui ne pourront pas être corrigés sera nécessaire.

3.2.3 Bilan

Lorsque il est nécessaire d'assurer un haut niveau d'intégrité sur chaque message, la seule possibilité pour augmenter le pouvoir de détection d'un code détecteur d'erreurs est d'augmenter le nombre de bits de contrôle. Quelle que soit la mise en œuvre utilisée, dans le cas de SCC, la taille des messages étant relativement faible (par ex., 100 bits dans pour notre cas d'étude), cela à un impact non négligeable sur le rendement du code, et peut avoir des répercussions importantes en temps de calcul. De plus, si la protection est mise en défaut sur un message, elle risque fort de l'être sur les messages suivants, dans la mesure où la même erreur a une forte probabilité de se reproduire. C'est notamment le cas d'une défaillance fonctionnelle, liée à la mémorisation ou au traitement des données reçues par les nœuds intermédiaires (erreurs multiples répétitives).

Or, rappelons que, dans le cas des systèmes considérés (systèmes à dynamique lente), il n'est pas nécessaire de garantir un haut niveau d'intégrité sur chaque message, mais plutôt sur un lot de messages. En tirant profit de cette caractéristique, nous allons proposer une fonction de protection originale qui ne présente pas les inconvénients de l'augmentation du nombre de bits de contrôle décrits précédemment, tout en garantissant un niveau d'intégrité tout aussi important pour le type d'applications considérées.

Et en plus d'assurer un haut niveau d'intégrité sur un lot de messages, il ne faudra pas oublier que la solution proposée devra également satisfaire les fortes contraintes temps réel des systèmes de communication concernés par notre cas d'étude [Youssef *et al.* 2005].

3.3 Solution proposée

Dans les sections suivantes, nous décrivons le principe général de la solution proposée. Puis, nous discutons de son apport vis-à-vis des erreurs multiples répétitives induites par des fautes de type "collage de bit" ou "erreur d'adressage mémoire" des nœuds intermédiaires.

3.3.1 Principe général

L'idée consiste, pour les systèmes à dynamique lente, à ne pas mettre un pouvoir de détection important sur *chaque message*, mais sur *une série de messages*. La solution propose que l'émetteur fasse évoluer (change) la fonction de contrôle à chaque message émis, selon une loi commune aux émetteurs et récepteurs. Le nœud émetteur utilise *m* fonctions *F1*, *F2*, … , *Fm*, de contrôle d'erreur, générant toutes le même nombre de bits de contrôle. Nous appelons cette technique *la fonction de contrôle d'erreur évolutive*. Plusieurs types de lois de changement cyclique des fonctions de contrôle sont envisageables (cf. figure 3.6) [Youssef *et al.* 2004].

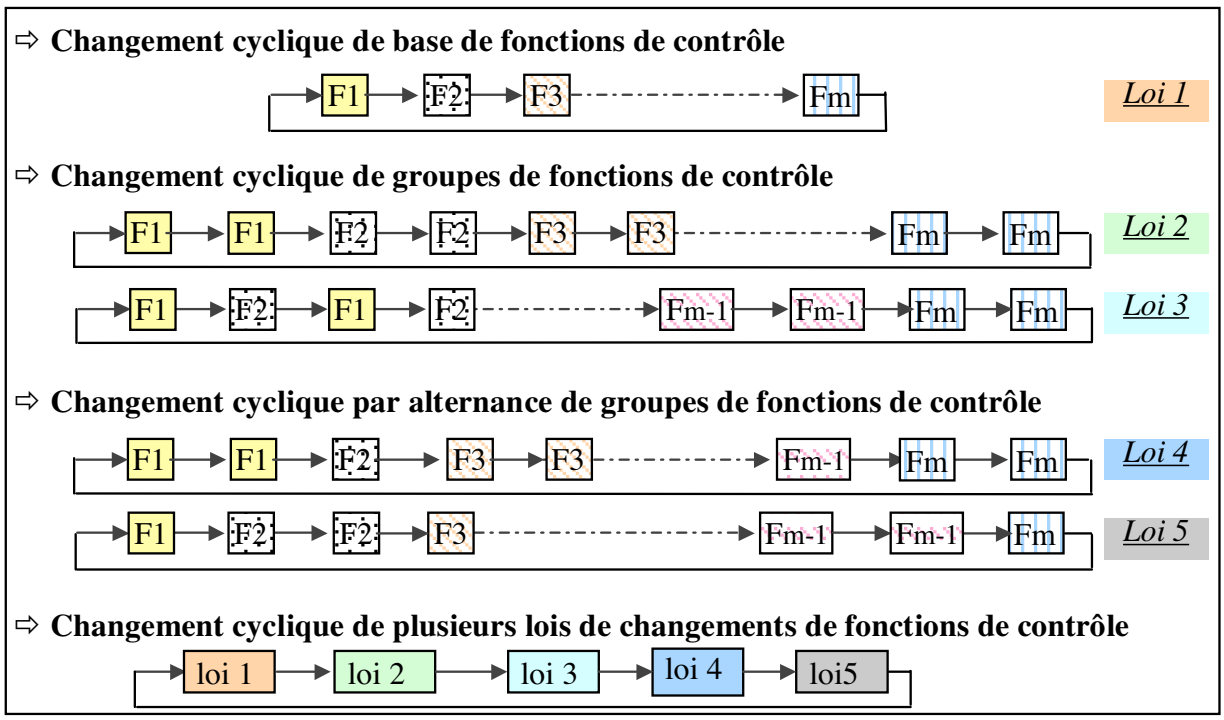


Figure 3.6 - Exemples de lois de changement cyclique de fonctions de contrôle

En raisonnant sur un lot de messages erronés, l'impact de la loi de changement cyclique des fonctions de contrôle diffère d'une loi à une autre, à la fois sur le nombre et les positions des messages erronés détectés dans ce lot. Cette différence dépend, d'une part, de la façon avec laquelle chaque loi alterne les fonctions de contrôle, et d'autre part, des pouvoirs de détection d'erreurs de chacune de ces fonctions.

Pour décrire simplement la fonction de contrôle évolutive, nous prendrons désormais le cas de la loi de changement cyclique de base *F1, F2, …, Fm, F1, F2, … , Fm,* etc., (*loi 1*, figure 3.6).

Pour que l'apport de la fonction de contrôle évolutive, en termes de pouvoir de détection d'erreurs, soit optimal, il faut que les fonctions de contrôle utilisées aient des pouvoirs de détection complémentaires et donc cumulables. En effet, il faut faire en sorte que la probabilité de non détection d'un message erroné par une fonction de contrôle, sachant qu'il n'a pas été détecté par la fonction précédente, soit très faible, voire nulle.

Il est important de noter que, dans les systèmes que nous considérons, si les nœuds intermédiaires ont un rôle applicatif, ils devraient alors utiliser la technique de changement cyclique de fonctions de contrôle pour vérifier l'intégrité des données reçues avant leur traitement. Si les nœuds intermédiaires régénèrent un nouveau champ de contrôle relatif aux données traitées, il n'est pas alors possible de garantir un haut niveau d'intégrité des données applicatives transmises de bout en bout du système de communication, ceci dans la mesure où les données peuvent être altérées localement, et où le nouveau champ de contrôle prendrait en compte les nouvelles données. Ces données erronées transmises par les nœuds intermédiaires ne pourront pas être alors détectées par les nœuds récepteurs. Nous excluons donc pour l'instant tout rôle applicatif des nœuds intermédiaires.

Ci-après, nous illustrons l'apport de la fonction de contrôle évolutive en termes de pouvoirs de détection d'erreurs, en présence des deux modes de défaillances fonctionnelles des nœuds intermédiaires "collage de bits" et "erreurs d'adressage mémoire", pour lesquels l'étude des risques faite au chapitre 2 avait montré que les fonctions de contrôle d'erreur classiquement utilisées dans les couches basses d'un système de communication étaient insuffisantes, voire inopérantes, pour couvrir les risques dus à ces modes de défaillances, et qui donc ne permettaient pas, pour notre cas d'étude, de satisfaire les objectifs d'intégrité fixés.

3.3.2 Apport de la fonction de contrôle évolutive vis-à-vis du mode de défaillance "collage de bits"

Nous rappelons que ce type de défaillance fonctionnelle du tampon de stockage des données du nœud intermédiaire altère une suite de messages qui y sont mémorisés en collant certains de leurs bits à 1 ou à 0. Ainsi, du côté du nœud récepteur, ces altérations ont une probabilité P_F de ne pas être détectées par la fonction de contrôle F utilisée.

Considérons en plus le cas particulier d'un système de communication pour des échanges périodiques de messages d'état, donc avec une probabilité non négligeable qu'un nœud émetteur envoie sur plusieurs périodes le même message (cf. figure 3.7). Dans ce cas, vu que les messages reçus sont identiques, la probabilité qu'à la réception l'altération d'un message ne soit pas détectée par la fonction de contrôle *F* est de 1, sachant que l'altération du message précédent ne l'a pas été.

On retrouve le même le risque dans le cas où la différence entre plusieurs messages successifs porte seulement sur quelques bits (par exemple, les bits de poids faible), et que la défaillance fonctionnelle "collage de bits" du nœud intermédiaire porte sur les bits de poids forts.

Appliquons maintenant la fonction de contrôle évolutive à ces deux derniers cas, pour montrer l'amélioration apportée en termes de pouvoir de détection d'erreurs. Nous proposons d'utiliser *m* fonctions de contrôles et de les changer cycliquement, comme précisé plus haut, suivant la *loi 1* : *F1, F2, … , Fm, F1, F2, … , Fm, ...*

Au niveau du nœud récepteur, pour un premier message erroné et non détecté par la fonction de contrôle correspondante *F1*, les probabilités de non détection des messages erronés suivants reçus sont largement inférieures à 1 (cf. figure 3.7). En effet, si les *m* fonctions de contrôle utilisées ont des pouvoirs de détections d'erreurs cumulables, alors la probabilité qu'un message erroné ne soit pas détecté par la fonction de contrôle *Fi*, sachant que la fonction de contrôle *Fj* ne l'a pas détecté, est au moins inférieure à 1 (avec $1 \leq j < i \leq m$).

En fait, cette probabilité est d'autant plus faible que la complémentarité entre les pouvoirs de détection d'erreurs des fonctions de contrôle utilisées est grande.

Les moyens pour atteindre cette complémentarité dépendent des caractéristiques des fonctions de contrôle utilisées. Dans le chapitre 4, nous développons une technique garantissant une complémentarité maximale entre les fonctions de contrôle d'erreur avec des codes CRC.

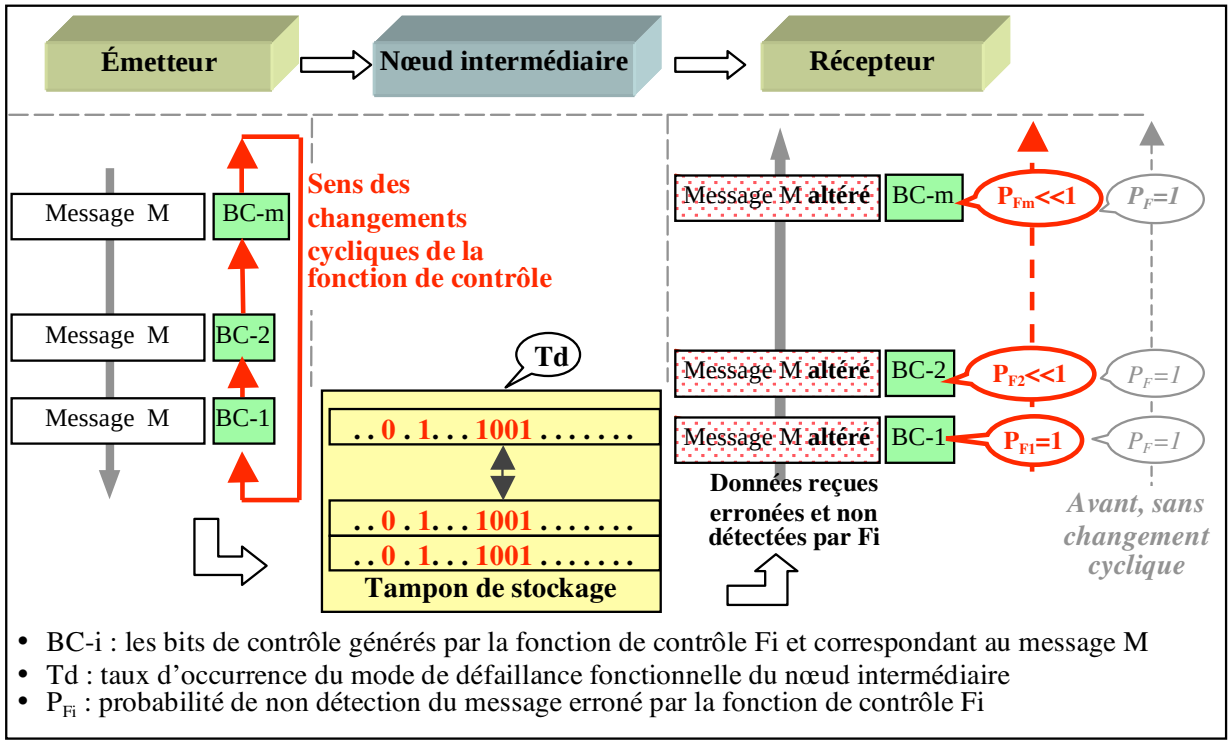


Figure 3.7 - Apport de la fonction de contrôle d'erreur évolutive (collage de bits)

Illustrons maintenant l'apport de la fonction de contrôle évolutive en présence du deuxième mode de défaillance fonctionnelle du nœud intermédiaire : "erreur d'adressage mémoire".

3.3.3 Apport de la fonction de contrôle évolutive vis-à-vis du mode de défaillance "erreur d'adressage mémoire"

Nous avons vu que le nœud intermédiaire peut présenter des risques vis-à-vis du non rafraîchissement quand sa zone mémoire n'est plus accessible en écriture ou en cas d'une erreur d'adressage mémoire. Dans ces cas, le contrôleur du nœud intermédiaire peut utiliser une zone mémoire pouvant contenir des messages corrects mais qui d'un point de vue temporel sont périmés.

En l'absence de mécanismes tels que celui évoqué au paragraphe 3.1.3, l'utilisation d'une seule fonction de contrôle d'erreur entre les nœuds émetteur et récepteur ne permet pas à ce dernier de détecter la péremption temporelle des messages, vu que les bits de contrôle générés par la fonction de contrôle d'erreur correspondent aux données reçues, bien que celles-ci soient périmées. Par contre, en changeant cycliquement de fonctions de contrôle, le nœud récepteur recalcule les bits de contrôle sur la base des mêmes données que dans le message précédent, mais en utilisant une fonction de contrôle différente de celle qui avaient été utilisée pour ce message précédent (cf. figure 3.8). Ainsi, même en l'absence de mécanisme spécifique de datation des messages, l'utilisation d'une fonction de contrôle évolutive garantit, avec une certaine probabilité, que les bits de contrôle calculés ne correspondent pas à ceux reçus, conduisant ainsi à la détection implicite par le nœud récepteur de messages périmés.

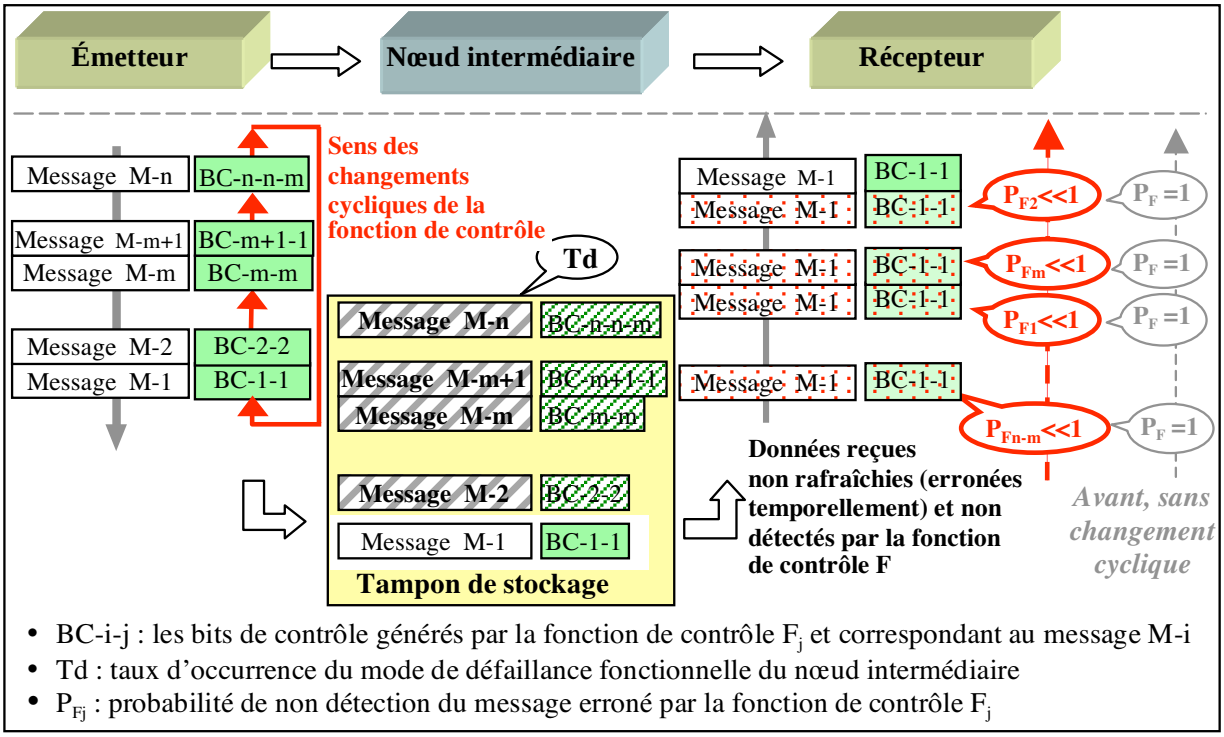


Figure 3.8 - Apport de la solution proposée en termes de détection de la péremption temporelle des messages

3.4 Application de la fonction de contrôle évolutive au système de communication pour les CDV

Nous avons considéré que les systèmes de CDV sont à dynamique lente vu que le temps de traitement d'une commande par la surface de contrôle est plusieurs fois supérieur à la période de son rafraîchissement. En effet, la vitesse maximale de déplacement d'une surface de contrôle étant de 50 degrés/s, elle pourrait recevoir une commande (message d'état) de se déplacer de sa position initiale (ex. : 0 degré par rapport à l'aile de l'avion) à une position de 50 degrés, ce qui nécessite un temps minimum d'exécution de 1 seconde. Ce temps est 100 fois plus grand que la période de rafraîchissement (10 ms) de la surface de contrôle par les commandes du calculateur.

Quant à l'architecture simplifiée de communication proposée pour les systèmes de CDV, décrite au chapitre 2, elle se base sur l'interconnexion de nœuds intermédiaires entre le calculateur et le nœud responsable d'activer la surface de contrôle et de transmettre des retours capteurs. Par abus de langage, ce dernier nœud est assimilé à la surface de contrôle dont il gère l'activation.

Par ailleurs, nous avons montré que, dans une telle architecture, la détection des altérations pouvant survenir lors de la transmission de la commande du calculateur à la surface de contrôle, ou pendant le retour de position de cette surface, ne pouvait pas reposer sur des unités CRC standard mises en œuvre matériellement au niveau de chaque nœud du système, y compris les nœuds intermédiaires. En effet, l'étude des risques développé au chapitre 2 a montré que, dans le cas de certaines défaillances fonctionnelles des nœuds intermédiaires, de telles unités CRC seraient insuffisantes, voire inopérantes, en termes de pouvoir de détection d'erreurs pour satisfaire les objectifs d'intégrité visés dans le cadre de notre cas d'étude.

La mise en œuvre d'une fonction de contrôle évolutive entre les couches application du nœud émetteur (calculateur) et du nœud récepteur (surface de contrôle) permet d'augmenter ce pouvoir de détection d'erreurs. Mais, il convient de vérifier si cela permet d'atteindre l'objectif d'intégrité de haut niveau qui, pour les systèmes de CDV, est de garantir un taux d'embarquement d'une surface de contrôle inférieur à 10^{-9}/heure de vol.

Cet embarquement peut être dû à la non détection, et donc au traitement, d'un nombre *X* de commandes erronées dans un lot de *N* commandes reçues par la surface de contrôle. Il dépend aussi de la stratégie de recouvrement d'erreur utilisée suite à la détection de commandes erronées.

Le recouvrement se base sur l'utilisation de moyens de commande-contrôle redondants, de manière à ce que le système de CDV continue à fonctionner correctement en présence d'erreurs (cf. chapitre 1). Il peut consister, suite à la détection de *Z* commandes erronées par la surface de contrôle, à désactiver toute la voie de commande et à basculer vers une autre voie pour commander la surface de contrôle. Il peut aussi consister à désactiver la surface de contrôle, et à commander les surfaces de contrôle restantes de sorte à compenser cette désactivation.

Dans tous les cas, la stratégie de recouvrement vise à éviter l'embarquement des surfaces de contrôle pour satisfaire l'objectif d'intégrité de haut niveau. Toutefois, l'utilisation intempestive d'un recouvrement, surtout dans le cas où il aboutit à la désactivation d'une ou de plusieurs surfaces de contrôle, nuît à la disponibilité du système de CDV. En effet, si la désactivation des surfaces de contrôle s'effectue dès la détection d'une seule commande erronée, le risque de l'embarquement devient très faible, voire nul. Par contre, les surfaces de contrôle seront beaucoup moins disponibles, et donc le système de CDV le sera aussi.

Dans ce qui suit, nous décrivons plus en détail la relation entre la stratégie de recouvrement d'erreur utilisée et le risque d'embarquement de la surface de contrôle.

3.4.1 Stratégie de recouvrement et risque d'embarquement

En nous basant sur un cas de défaillance particulier, nous décrivons d'abord la relation existant entre la stratégie de recouvrement d'erreur et le risque d'embarquement de la surface de contrôle. Par la suite, en utilisant le même exemple, nous présentons les différentes analyses qui, d'une part, montrent la corrélation existant entre la loi de changement cyclique de fonctions de contrôle et la stratégie de recouvrement d'erreur utilisée et, d'autre part, permettent de choisir la stratégie de recouvrement d'erreur la plus appropriée.

En se référant à l'architecture de communication de base présentée au chapitre 2, supposons que l'occurrence d'une défaillance fonctionnelle du nœud intermédiaire (par exemple, erreur d'adressage mémoire) entraîne la transmission répétitive de commandes erronées. Deux cas peuvent alors être envisagés au niveau de la surface de contrôle :

1) Les commandes erronées reçues pas la surface de contrôle sont non détectées par la fonction de contrôle d'erreur utilisée, et sont donc traitées. Dans ce cas, si la valeur des commandes erronées s'écarte trop de la valeur réelle, le traitement répétitif de ces commandes par la surface de contrôle entraîne son embarquement.
2) Les commandes erronées reçues pas la surface de contrôle sont détectées et donc rejetées. Dans ce cas, comme pour les protocoles ARINC, la surface de contrôle utilise la dernière commande jugée correcte par sa fonction de contrôle, mais qui peut être en réalité une commande erronée et non détectée.

Pour analyser la relation entre la stratégie de recouvrement et le risque d'embarquement de la surface de contrôle (cf. figure 3.9), nous utilisons : 1) une variable de supervision ***J***, qui s'incrémente à chaque non détection et donc traitement d'une commande erronée reçue par la surface de contrôle, jusqu'à atteindre la valeur seuil ***L1***, synonyme d'embarquement de la surface de contrôle, et 2) une variable ***I*** qui s'incrémente à chaque détection et par suite rejet d'une commande erronée reçue par la surface de contrôle, jusqu'à atteindre la valeur seuil ***L2***, synonyme de déclenchement de la stratégie de recouvrement, qui vise à éviter l'embarquement de la surface de contrôle.

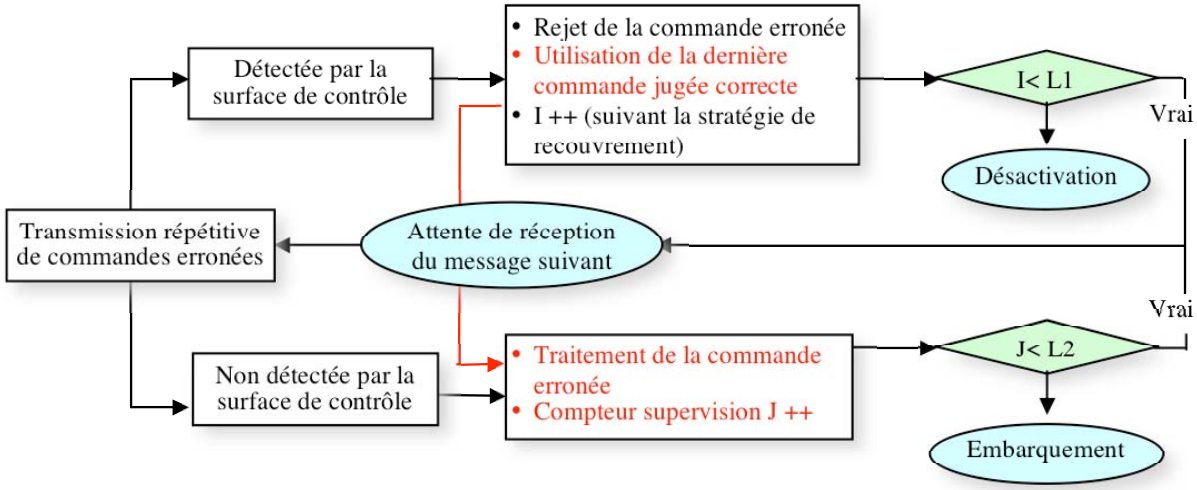


Figure 3.9 - Stratégie de recouvrement et embarquement des surfaces de contrôle

Les valeurs seuil *L1* et *L2* sont corrélées. En effet, plus la valeur de *L1* est élevée, plus la surface de contrôle tolère le traitement de commandes erronées sans atteindre l'embarquement, et donc plus il est possible d'augmenter la valeur seuil *L2*, évitant ainsi le déclenchement de la stratégie de recouvrement d'erreur qui pourrait conduire à la désactivation intempestive de la surface de contrôle, et donc nuire à la disponibilité du système de CDV.

Il faut donc fixer les valeurs seuil *L1* et *L2* de sorte que la surface de contrôle puisse, à la détection de commandes erronées, rester le plus longtemps possible disponible sans qu'elle embarque. La détermination de ces valeurs se fait par le choix d'une stratégie de recouvrement appropriée.

3.4.2 Choix de la stratégie de recouvrement

Cette section présente une analyse comparative entre plusieurs stratégies de recouvrement, ceci en se basant sur le cas de défaillance décrit à la section précédente. Plus particulièrement, nous nous basons sur la situation d'embarquement de la surface de contrôle, suite au traitement répétitif pendant un temps suffisamment long (100 ms correspondant à une série de 10 rafraîchissements de 10 ms) de commandes erronées et non détectées, et dont la valeur s'écarte trop de la valeur réelle

En se référant à l'architecture de communication de base pour les systèmes de CDV, la probabilité d'occurrence de ce type d'embarquement correspond à la probabilité d'occurrence *P(ND)* d'une défaillance fonctionnelle d'un nœud intermédiaire provoquant la transmission des commandes erronées, multipliée par la probabilité *P1* de leur non détection par la fonction de contrôle d'erreur utilisée *F1* (cf. figure 3.10).

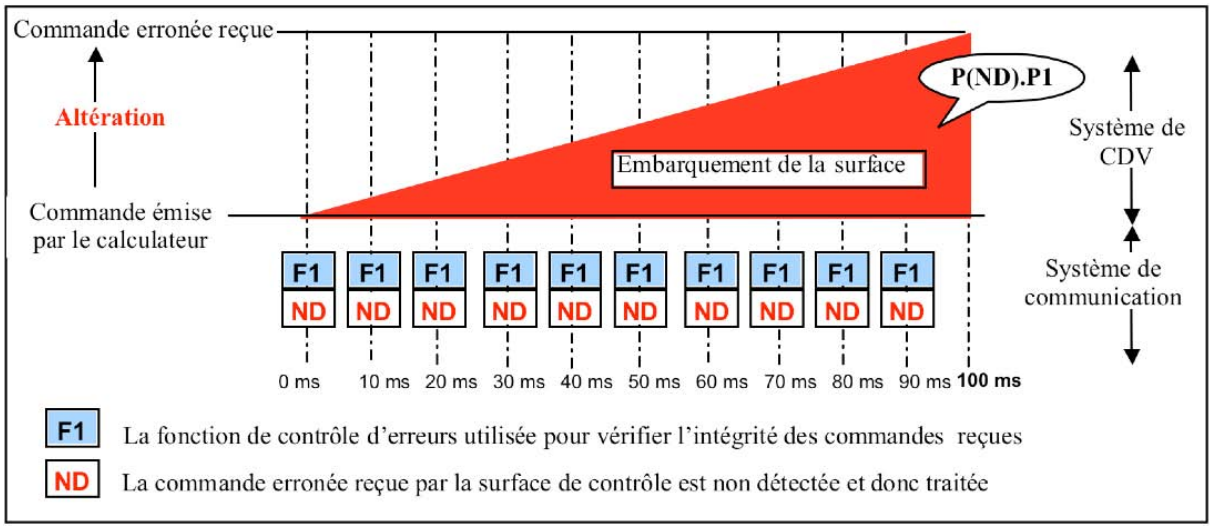


Figure 3.10 - Illustration d'un cas d'embarquement de surface de contrôle

L'application de la technique de changement cyclique de fonctions de contrôle d'erreur entre le calculateur et la surface de contrôle augmente le pouvoir de détection des commandes erronées reçues par la surface de contrôle. Par exemple, en utilisant un changement cyclique de deux fonctions de contrôle *F1* et *F2* ayant des pouvoirs de détection cumulables, l'apport en termes de détection de commandes erronées sera proche de 50%. En utilisant un changement cyclique sur 3 polynômes générateurs, cet apport sera proche de 66,66%. etc.…(cf. § 4.4.2)

En présence du mode de défaillance considéré, appliquons maintenant différentes stratégies de recouvrement pour choisir celle qui offre, d'une part, le plus grand pouvoir de détection d'erreurs et, d'autre part, la meilleure disponibilité des surfaces de contrôle tout en évitant leur embarquement. Pour cela, nous définissons les variables suivantes :

- La variable entière $\boldsymbol{X_{max}}$ représentant le nombre maximum de commandes erronées détectées que la surface peut tolérer avant le déclenchement de la stratégie de recouvrement (basculement vers une autre voie de commande et/ou désactivation de la surface de contrôle).
- La variable booléenne ***Conf*** : elle vaut 1 si on raisonne sur des détections consécutives des commandes erronées reçues (stratégie avec confirmation), et elle vaut 0 sinon (stratégie sans confirmation). Dans ce dernier cas, on raisonne plutôt par rapport au nombre de détections de commandes erronées dans un lot de commandes reçues. Pour un rafraîchissement périodique des surfaces de contrôle tous les 10 ms et vu que l'embarquement d'une surface de contrôle est jugé sur 100 ms, ce lot représente une série de 10 commandes reçues par la surface de contrôle.

À titre d'exemple, en appliquant les définitions ci-dessus pour $X_{max} = 2$, $Conf = 1$, le recouvrement est déclenché au bout de trois détections consécutives de commandes erronées par la surface de contrôle, ce qui correspond à trois non rafraîchissements consécutifs.

Ci-après nous comparons les deux types de stratégies de recouvrement, avec et sans confirmation. En première approche, nous choisissons les cas particuliers d'une stratégie avec confirmation ($X_{max} = 2$, $Conf = 1$), que l'on notera ***S2***, et d'une stratégie sans confirmation ($X_{max} = 2$, $Conf = 0$), que l'on notera ***S2'***. La comparaison se fait en trois étapes en augmentant progressivement à chacune d'elles le nombre de fonctions de contrôle utilisées (cf. figure 3.11).

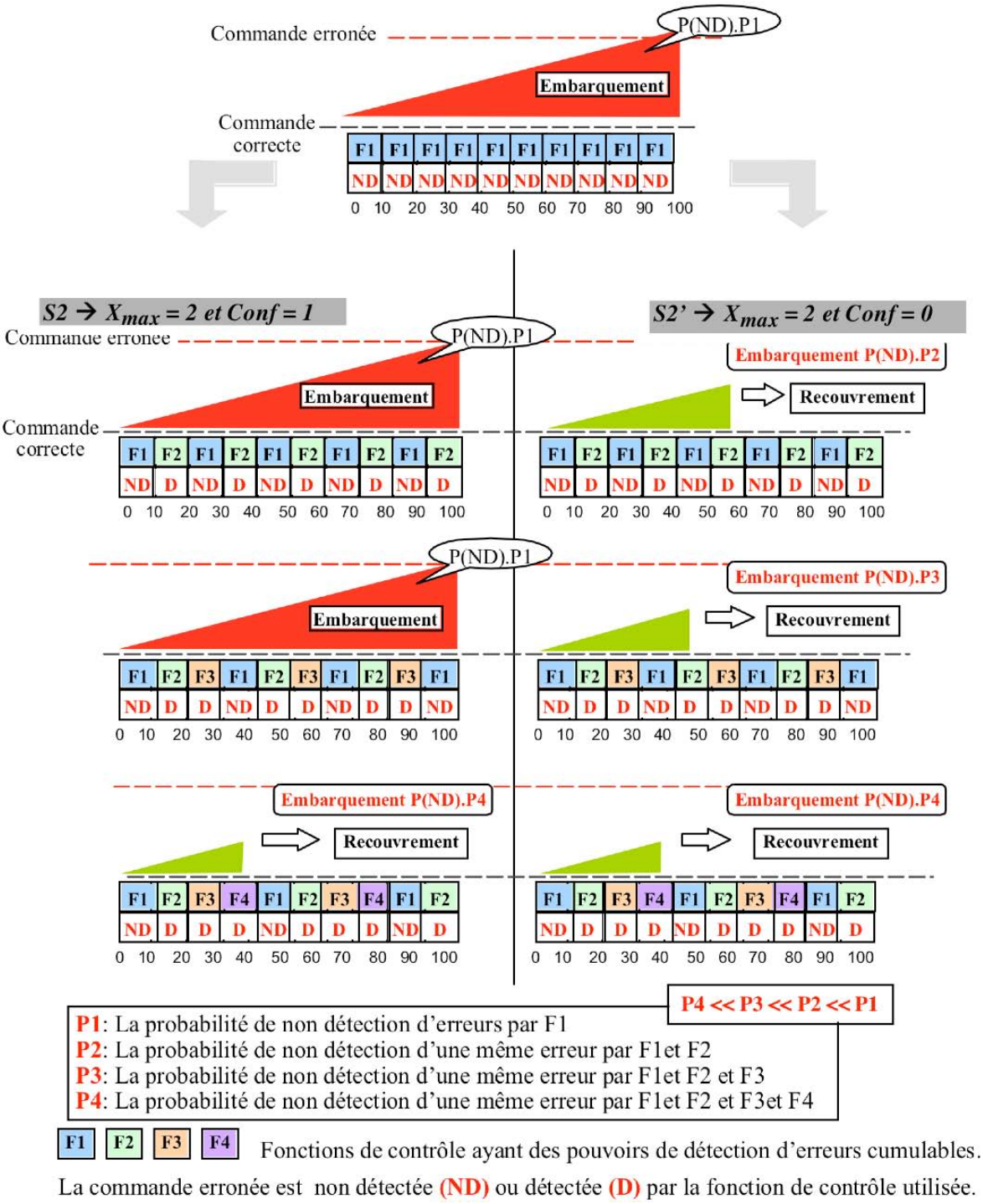


Figure 3.11 - Analyse comparative entre S2 et S2'

Le changement cyclique des différentes fonctions de contrôle s'effectue selon la *loi 1* de la figure 3.6. Ainsi, à la première étape de la comparaison des deux stratégies de recouvrement, on utilise le changement cyclique (*F1-F2-F1-F2---*), à la deuxième étape, on utilise (*F1-F2-F3-F1-F2-F3---*) et à la troisième étape, on utilise (*F1-F2-F3-F4-F1-F2-F3-F4--*).

Un des critères utilisés pour comparer les deux types de stratégies de recouvrement est le "temps pré-recouvrement", que nous définissons comme le temps qui s'écoule entre le moment de la première détection d'une commande erronée reçue par la surface de contrôle et le moment de déclenchement du recouvrement.

Nous cherchons à assurer un "temps pré-recouvrement" qui soit le plus faible possible pour, d'une part, augmenter le pouvoir de détection des commandes erronées par la fonction de contrôle évolutive, et d'autre part, optimiser le temps de réaction du système de CDV.

En effet, plus la réaction du système à l'occurrence d'une défaillance (celle du nœud intermédiaire par exemple) est rapide, plus il serait possible d'améliorer la stratégie de recouvrement utilisée pour qu'elle puisse tolérer un nombre plus important de détections de commandes erronées. Dans ce cas, la désactivation intempestive des surfaces de contrôle pourrait être diminuée et la disponibilité du système de CDV se trouverait ainsi améliorée.

La première constatation est que, plus le nombre de fonctions de contrôle est important, plus le "temps pré-recouvrement" est faible. En effet, sur la base de la stratégie *S2'*, ce temps passe de 50 ms avec 2 fonctions de contrôle à 40 ms dans le cas de 3 fonctions de contrôle, et à 30 ms en utilisant 4 fonctions de contrôle. Toutefois, il est important de noter qu'avec 4 (ou plus) fonctions de contrôle, le "temps pré-recouvrement" de 30 ms reste inchangé et ceci pour les deux stratégies utilisées. Ceci explique notre choix de s'arrêter dans cette analyse comparative, et dans celles qui suivront, au changement cyclique de 4 fonctions de contrôle.

La deuxième constatation, et qui sera déterminante dans le choix de la stratégie de recouvrement, est qu'en utilisant *S2'*, la probabilité d'embarquement de la surface de contrôle est déjà diminuée dès l'utilisation de 2 fonctions de contrôle, par rapport à *S2*, avec qui il faut utiliser au minimum 4 fonctions de contrôle pour diminuer la probabilité d'embarquement, et atteindre un temps de "pré-recouvrement" égal à celui obtenu en utilisant *S2'*.

Donc, et suite à cette première analyse comparative, la stratégie *S2'* s'avère meilleure que *S2*, du point de vue de la probabilité d'embarquement des surfaces de contrôle, vu qu'elle nécessite moins de fonctions de contrôle que *S2*.

Afin de confirmer ce premier résultat, nous étendons ci-après l'analyse comparative aux stratégies *S1* ($X_{max} = 1$, $Conf = 1$) *S1'* ($X_{max} = 0$, $Conf = 0$), *S3* et *S3'*, *S4* et *S4'*. Nous présentons ci-après des comparaisons deux à deux entre les stratégies de recouvrement avec et sans confirmation, afin d'illustrer l'impact du choix d'une stratégie sur les états de la surface de contrôle : embarquement ou phase de recouvrement, et, dans ce dernier cas, nous précisons le temps pré-recouvrement (cf. figure 3.12).

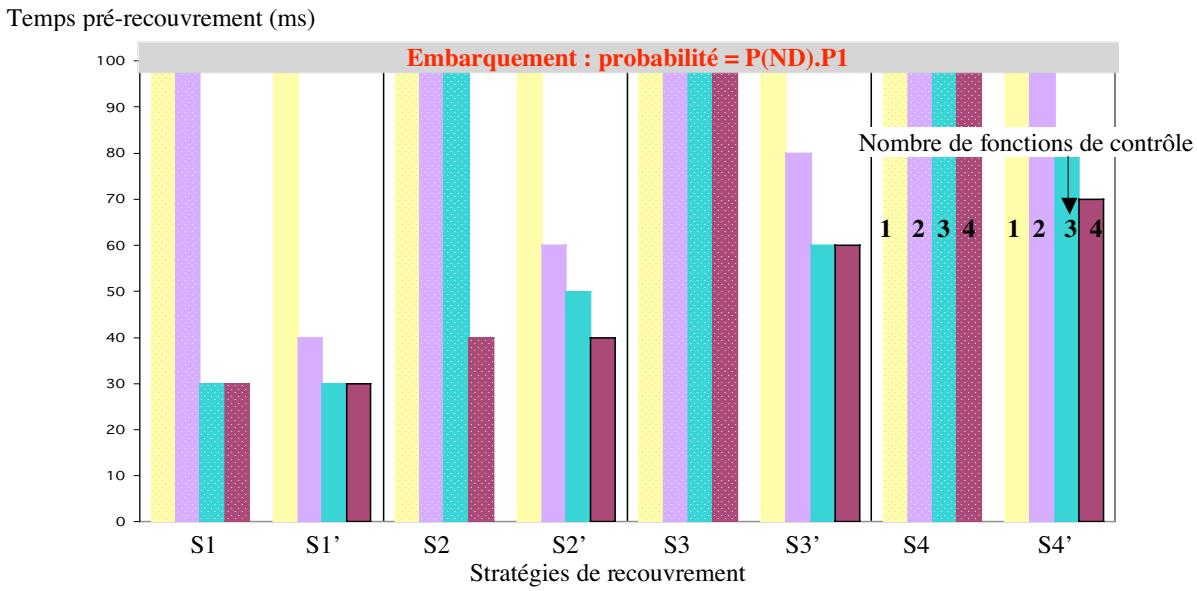


Figure 3.12 - Analyse comparative entre les stratégies de recouvrement

Les résultats de cette analyse comparative confirment le premier résultat. En effet, toutes les stratégies de recouvrement sans confirmation utilisées assurent le "temps pré-recouvrement" le plus faible. Et l'occurrence de l'embarquement de la surface de contrôle en utilisant ces stratégies est 7 fois moins fréquente qu'avec les stratégies avec confirmation (14 occurrences d'embarquement avec *S2*, *S3* et *S4*, pour seulement 2 occurrences avec *S2'*, *S3'* et *S4'*).

Ces résultats s'expliquent par le fait que la probabilité d'avoir *x1* détections de commandes erronées par la fonction de contrôle évolutive est supérieure à celle d'avoir *x2* détections consécutives de commandes erronées, et ceci même dans le cas où *x2* est inférieur à *x1*. En effet, d'après la figure 3.12, on remarque qu'en utilisant un changement cyclique de 2 fonctions de contrôle, et en se basant sur la stratégie *S2* (donc $x2 = 3$), c'est-à-dire un recouvrement déclenché au bout de la troisième détection consécutive de commandes erronées, la probabilité d'embarquement de la surface de contrôle n'est pas diminuée. Par contre, en se basant sur la stratégie sans confirmation *S3'* (donc $x1 = 4$), et en utilisant le même nombre de fonctions de contrôle, la probabilité d'embarquement est diminuée.

Au vu de ces résultats, nous pouvons affirmer que les stratégies de recouvrement sans confirmation sont les plus appropriées pour deux raisons. La première est la garantie d'une probabilité d'embarquement des surfaces de contrôle la plus faible. Cette caractéristique est d'une importance capitale vu que l'objectif d'intégrité de haut niveau de l'étude est d'assurer un taux d'embarquement des surfaces de contrôle qui soit inférieur à 10^{-9}/h. La deuxième raison est la garantie du temps pré-recouvrement le plus faible, et donc la possibilité d'offrir une meilleure disponibilité du système de CDV en présence de défaillances.

Toutefois, il est important de noter que l'impact d'une stratégie de recouvrement sur l'embarquement des surfaces de contrôle diffère selon la loi de changement cyclique de fonctions de contrôle utilisée. Le but de la section suivante est d'illustrer, puis de justifier cette corrélation, en se basant sur l'exemple de deux lois de changements cycliques de fonctions de contrôle décrits à la figure 3.6 (*loi 1* et *loi 2*). Rappelons que ces lois diffèrent selon la façon dont elles alternent les fonctions de contrôle, et que cette caractéristique a un impact sur le nombre et la consécutivité des détections des commandes erronées dans un lot de commandes.

3.4.3 Loi de changement cyclique et stratégie de recouvrement

Pour chacune des stratégies de recouvrement avec et sans confirmation, nous présentons dans cette section une analyse comparative entre deux lois de changements cycliques de fonctions de contrôle (la *loi 1* et la *loi 2*). Concrètement, en utilisant un changement cyclique de 4 fonctions de contrôle, on aurait l'alternance suivante (*F1-F2-F3-F4-F1-F2-F3-F4*---) pour la *loi 1*, et (*F1-F1-F2-F2-F3-F3-F4-F4-F1-F1*---) pour la *loi 2*.

Cette analyse a pour objectif principal de dégager les caractéristiques génériques que devraient avoir les lois de changements cycliques de fonctions de contrôle pour garantir, suivant la stratégie de recouvrement utilisée, le meilleur apport en termes de diminution de la probabilité d'embarquement. Cela s'explique par le fait que la technique de changement cyclique sur laquelle se base la *loi 2* présente une série de plusieurs fonctions de contrôle redondantes (chacune est utilisée 2 fois de suite), ce qui permet d'utiliser entre deux fonctions de contrôle *F1* consécutives, une série de fonctions plus importante que celle présentée par la *loi 1*.

Vu que le cas d'analyse considéré est la réception par la surface de contrôle de commandes erronées non détectées par *F1*, et puisque les fonctions de contrôle d'erreur utilisées ont des pouvoirs de détection d'erreurs complémentaires, notamment avec ceux de *F1*, il est donc plus probable d'avoir une confirmation de la détection des commandes erronées en utilisant la *loi 2*.

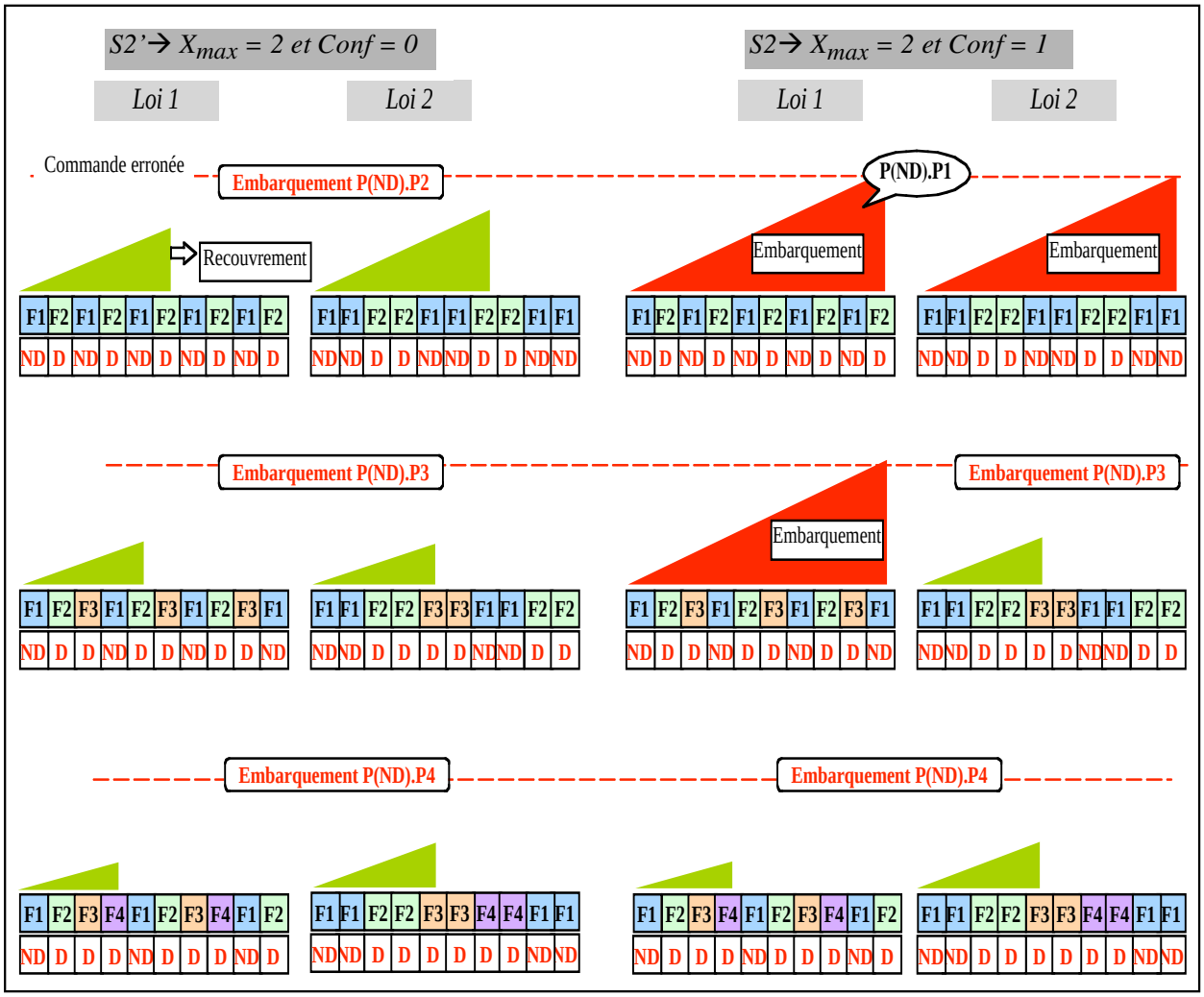


Figure 3.13 - Analyse comparative entre la *loi 1* et la *loi 2*

Dans le cas de la stratégie de recouvrement sans confirmation *S2'*, les deux lois utilisées permettent de diminuer la probabilité d'embarquement en utilisant seulement deux fonctions de contrôle d'erreur. Par contre, du point de vue pouvoir de détection d'erreurs, la *loi 1* est meilleure que la *loi 2* vu qu'elle assure le temps pré-recouvrement le plus faible.

Ce dernier résultat s'explique par le fait que la technique de changement cyclique sur laquelle se base la *loi 1* présente une alternance plus rapide entre les fonctions de contrôle d'erreur utilisées. Concrètement, dans l'intervalle de temps des 100 ms, la surface de contrôle reçoit 10 commandes erronées et non détectées par la fonction de contrôle *F1*, dont un plus grand nombre seraient détectées, et plus rapidement en utilisant une alternance plus fréquente entre les autres fonctions de contrôle (*F2-F3-F4*).

Nous pouvons donc affirmer que plus la fréquence d'alternance entre les fonctions de contrôle d'erreur utilisées est grande, plus leur pouvoir de détection cumulé est important. Dit autrement, plus l'évolution de la fonction de contrôle évolutive est importante, plus elle offre un pouvoir de détection important et rapide (temps de pré-recouvrement faible).

Nous venons de décrire la fonction de contrôle d'erreur évolutive, puis nous l'avons validée par rapport aux objectifs d'intégrité fixés dans le cas général, puis dans le cas du système de CDV, ceci en raisonnant sur une mise en œuvre au niveau de la couche "application".

En fait, la technique de changement cyclique de fonctions de contrôle, ainsi que les techniques de contrôle des erreurs en valeur, d'adressage et temporelles décrites au début de ce chapitre, peuvent être mises en œuvre au niveau des couches logicielles inférieures d'un système de communication. En référence au modèle OSI, il s'agit des couches de gestion de l'application.

Le but de la section suivante est de redéfinir le cadre dans lequel s'inscrit notre étude, en précisant, d'une part, les couches protocolaires, autres que la couche application, où l'on pourrait mettre en œuvre les techniques de protection proposées et, d'autre part, la limite à partir de laquelle l'apport en termes d'intégrité de ces techniques n'est plus garanti.

3.5 Généralisation du cadre de l'étude de l'intégrité

Pour une architecture de communication avec plusieurs niveaux de nœuds intermédiaires, nous pouvons dire que la garantie d'un apport optimal de la fonction de contrôle évolutive au niveau de la couche logicielle où elle est mise en œuvre dépend de la propriété suivante :

> *"Les fonctionnalités de chaque nœud intermédiaire au niveau de cette couche, si elles existent, ne doivent pas présenter des techniques de contrôle d'erreur".*

Prenons l'exemple de la figure 3.14, où les données transitent au travers de deux nœuds intermédiaires pouvant effectuer du traitement logiciel, et dont le premier présente des fonctionnalités sur *i-1* couches et le second des fonctionnalités sur *i* couches. Par contre, les fonctionnalités des nœuds émetteurs et récepteurs s'étendent sur *z* couches (*z*>*i*).

Supposons également que toutes les couches logicielles possèdent des capacités de détection d'erreurs : au niveau de la couche application ($z^{ème}$ couche), des bits de contrôles sont générés par le nœud émetteur puis vérifiés, après transmission, par le nœud récepteur, en utilisant une fonction de contrôle *Fz* relative à la couche *z* ; au niveau de la couche *i+1*, c'est la fonction *Fi+1*, et ainsi de suite pour toutes les autres couches logicielles.

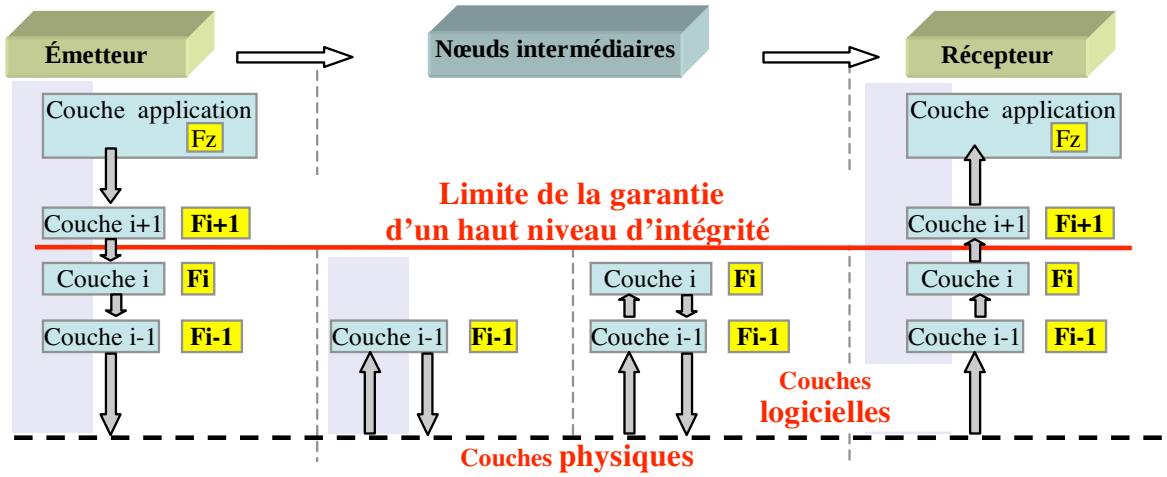


Figure 3.14 - Limite de la garantie d'un haut niveau d'intégrité

En se basant sur cet exemple, nous pouvons dire qu'il n'est possible de garantir un haut niveau d'intégrité que pour les données relatives aux couches *i+1* à *z*. En effet, le second nœud intermédiaire a la possibilité de vérifier l'intégrité des données qu'il reçoit, au niveau de la couche *i* (avec la fonction de contrôle *Fi*), de les traiter éventuellement, et puis de régénérer des bits de contrôle relatifs à ces données avant de les ré-émettre.

Dans ce cas, toute défaillance fonctionnelle de ce nœud peut se traduire par la transmission de données erronées, mais avec des bits de contrôle générés sur la base de ces données. Le nœud récepteur ne pourra donc pas détecter, en utilisant la fonction de contrôle *Fi*, ces données erronées. Ce raisonnement s'applique également pour toutes les autres couches logicielles en dessous de la couche *i*.

La fonction de contrôle d'erreur évolutive peut donc s'appliquer pour chacune des couches logicielles *i+1*, …, (cf. figure 3.14), en assurant un apport de pouvoir de détection d'erreurs.

Toutefois, les techniques sur lesquelles se base la fonction de contrôle d'erreur évolutive pourraient s'avérer complexes par rapport aux caractéristiques d'une couche, ce qui rend sa mise en œuvre difficile, voire impossible.

En général, le choix du nombre et du type des fonctions de contrôle d'erreur à utiliser, ainsi que le choix de leurs niveaux de mise en œuvre, dépendent essentiellement de quatre facteurs :

- la faisabilité technique,
- le coût en termes de temps de calcul et de débit de transmission,
- la confiance qu'il est possible d'accorder à l'intégrité des données transmises entre les différentes couches protocolaires,
- le niveau d'intégrité qui est visé au niveau de chaque couche [Totel *et al.* 1998].

En définitive, cette partie du chapitre visait à proposer une solution permettant d'augmenter le pouvoir de détection d'erreurs au niveau des couches logicielles situées en dessous de la limite de garantie d'un haut niveau d'intégrité (cf. figure 3.14). Dans ce cas, les nœuds intermédiaires peuvent vérifier l'intégrité de données reçues, les traiter, puis leur associer un nouveau champ de contrôle d'erreur. Maintenant, dans le cas où les nœuds intermédiaires auraient un rôle applicatif, nous allons voir quel peut être leur rôle sans que cela nuise à l'intégrité des communications.

3.6 Rôle applicatif des nœuds intermédiaires ?

Parmi les intérêts à utiliser des nappes de microsystèmes dans les SCC, et qui résulte de la possibilité de distribuer le traitement, il y a une granularité plus fine de la commande et une plus grande facilité pour tolérer la défaillance de microactionneurs via un traitement local. Il s'agit là d'opérations qui sont même masquées aux calculateurs de commande dans la mesure où : 1) ils n'ont pas à connaître les microsurfaces qui seront en fait en bout de la chaîne de commande, et 2) ils ne seront informés de défaillances dans les microactionneurs que si leur tolérance ne peut pas être assurée localement. Pour réaliser ces opérations, il est nécessaire d'allouer aux nœuds intermédiaires un rôle applicatif en lui donnant la possibilité d'agir sur la valeur de la commande qui sera transmise aux microactionneurs.

Il faut alors se demander comment autoriser un rôle applicatif aux nœuds intermédiaires sans que cela nuise à l'intégrité de la commande que l'on cherche à garantir de bout en bout, c'est-à-dire entre les calculateurs de commande (nœuds émetteurs) et les organes commandés (nœuds récepteurs) ? Cette section apporte des éléments de réponse à cette question.

Comme il est exclu que les nœuds intermédiaires soient basés sur des architectures à commande/surveillance, comme pour les calculateurs de commande (cf. figure 1.11), il n'est pas possible de garantir avec un haut niveau de sûreté que le traitement effectué au niveau du nœud intermédiaire soit correct. Il n'est donc pas possible d'autoriser un nœud intermédiaire à effectuer un changement important sur la valeur de la commande.

Pour des raisons de sécurité, nous avons donc prévu un champ *V* (initialement à 0) permettant au nœud intermédiaire d'agir (via une valeur signée) sur la commande qui, au final, sera appliquée ; l'assemblage de la valeur initiale et de la variation apportée par le nœud intermédiaire étant effectué au niveau du nœud récepteur.

Si l'on prévoit que la fonction de contrôle porte aussi sur le nouveau champ *V*, il ne faut pas utiliser localement un re-calcul complet des bits de contrôle, qui masquerait alors les erreurs qu'aurait pu introduire une défaillance du type collage de bits d'un tampon. On se retrouverait, en fait, avec le même problème que nous avons déjà évoqué au chapitre 2 concernant les insuffisances des fonctions de protection mises en œuvre au niveau de la couche physique.

La solution est alors d'utiliser une mise à jour incrémentale des bits de contrôle, technique que l'on utilise au niveau d'Internet [Braun & Waldvogel 2001, RFC 3385], où l'entête rajoutée aux messages par les couches basses change fréquemment. C'est notamment le cas du champ *Durée de vie* (ou *TTL* : *Time To Live*) qui indique la durée de vie maximale du paquet. Ce champ est décrémenté, de 1 à chaque passage dans un routeur, et du nombre de secondes passées dans la file d'attente. La mise à jour incrémentale vise à répondre à deux problèmes induits par un re-calcul complet : 1) le calcul du CRC sur tout le paquet est coûteux en temps, 2) le paquet n'est pas protégé contre les changements non désirés entre la dernière vérification et le re-calcul. En ce qui nous concerne, c'est surtout le deuxième aspect qui nous intéresse.

Le principe de mise à jour incrémentale du CRC repose sur le fait qu'un changement fréquent de l'entête ne nécessite pas un re-calcul complet en raison de la linéarité du codage CRC. Le principe que nous présentons s'applique donc à tout autre code linéaire. En notant I_1 et I_2 deux paquets d'information de même longueur, et $CRC(I_i)$ le CRC sur I_i, la linéarité du code conduit à $CRC(I_1+ I_2) = CRC(I_1) + CRC(I_2)$. Ainsi, pour un paquet composé d'un entête *h*, suivi d'un bloc de données *d* et du bloc *c* de CRC (noté *CRC(h, d)*), arrivant à un nœud qui transforme l'entête *h* en *h'*, la mise à jour consiste à ajouter à *c* la valeur *CRC(h'-h, 0)*, 0 étant un bloc de bits à 0 de longueur *d*.

L'utilisation que nous pourrions faire du principe de mise à jour incrémentale est décrite sur la figure 3.15, où le message émis est composé d'un bloc de données *D* contenant la commande, suivi d'un champ *V* destiné à accueillir la variation apportée à la commande, et d'un bloc de contrôle *F(D+V)* correspondant au champ de contrôle généré sur la base de *D* et de *V*. Le nœud intermédiaire modifie éventuellement la valeur du champ *V* et met à jour la valeur du champ de contrôle sans avoir besoin d'utiliser le champ *D*. Si une altération du champ *D* se produit au niveau du nœud intermédiaire, elle sera détectée au niveau du nœud récepteur.

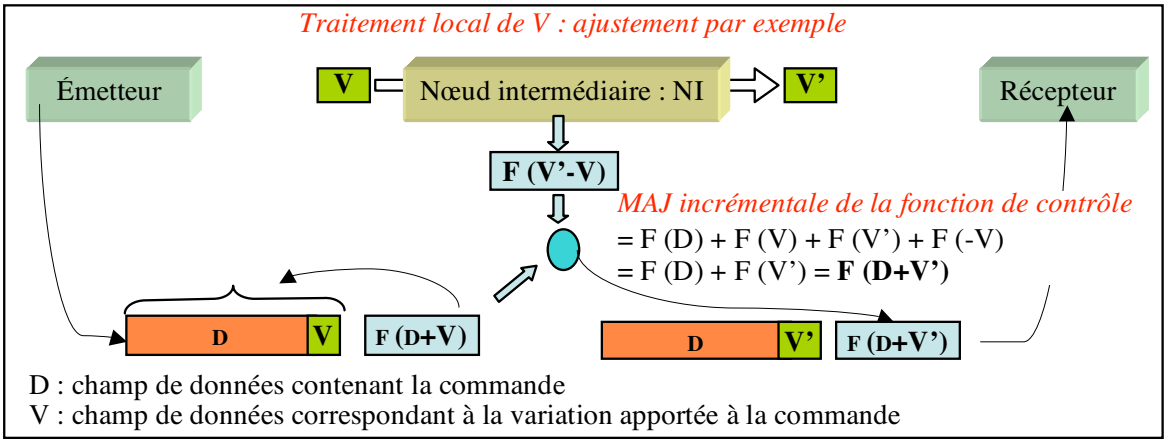


Figure 3.15 - Mise à jour incrémentale de la fonction de contrôle

Dans le cas de l'utilisation d'une fonction de contrôle évolutive, il faut bien sûr que la fonction de contrôle utilisée pour la mise à jour soit la même que celle utilisée lors de l'émission.

Le principe de la mise à jour incrémentale n'a finalement pas été retenu pour l'instant dans le cadre du cas d'étude. En effet, l'importance de la modification de la commande, autorisée localement au niveau d'un nœud intermédiaire, est pour l'instant faible, et une erreur sur cette modification est jugée non critique pour les surfaces contrôlées. Nous avons toutefois gardé le principe d'inclure dans le message un champ permettant aux nœuds intermédiaires d'avoir un rôle applicatif, mais sans prendre en compte ce champ dans le calcul des bits de contrôle.

3.7 Conclusion

Dans ce chapitre, pour faire face aux insuffisances des mécanismes de contrôle d'erreur usuellement mis en œuvre au niveau des composants de base d'un système de communication, nous avons tout d'abord proposé d'utiliser une fonction de contrôle de bout en bout faisant abstraction des couches physiques d'un système de communication. Ce principe est couramment retenu pour des systèmes avec un niveau de criticité comparable au nôtre. Nous avons néanmoins indiqué comment prendre en compte, au niveau d'une fonction de contrôle d'erreur globale, les trois types d'erreur à traiter pour répondre aux propriétés d'intégrité à satisfaire : les erreurs en valeur, les erreurs d'adressage et les erreurs temporelles.

Par la suite, pour atteindre le niveau d'intégrité visé tout en limitant autant que possible toute forme habituelle de redondance, nous avons proposé une solution originale qui tire profit du fait que, pour les SCC envisagés (systèmes à dynamique lente), l'intégrité est à considérer sur un lot de messages et non sur un seul message. Le principe de la fonction de contrôle évolutive que nous avons proposée consiste à coder chaque nouveau message en changeant (cycliquement par exemple) entre plusieurs fonctions de contrôle d'erreur ayant des pouvoirs de détection cumulatifs. Ainsi, dans le cas d'une erreur permanente, si elle n'est pas détectée par une fonction de contrôle, la probabilité est importante qu'elle soit détectée par un des différentes fonctions suivantes.

Concernant le changement cyclique des fonctions de contrôle, par le biais de l'exemple des CDV, nous avons montré l'impact du choix de la stratégie de recouvrement sur le taux d'occurrence de l'embarquement et la disponibilité du système de commande. Deux stratégies de désactivation de la surface de contrôle ont été comparées : 1) soit au bout de Z détections d'erreurs successives, 2) soit après la détection de Z erreurs dans un lot de N commandes.

L'application de la fonction de contrôle d'erreur évolutive au système de CDV a montré ensuite que, pour les modes de défaillances considérés pour les nœuds intermédiaires, il est possible de garantir un taux d'occurrence de l'événement redouté (embarquement des surfaces de contrôle) inférieur au seuil de criticité de 10^{-9}/h, satisfaisant ainsi l'objectif d'intégrité de haut niveau pour les systèmes de CDV.

Nous avons terminé ce chapitre en indiquant comment les nœuds intermédiaires pourraient avoir un rôle applicatif au niveau du système de commande-contrôle. La solution retenue pour le cas d'étude consiste à prévoir un champ spécial (non protégé) de données dans le message permettant au nœud intermédiaire d'agir de manière non critique sur la commande.

Le chapitre suivant décrit une mise en œuvre de la fonction de contrôle d'erreur évolutive à base de codes CRC. Il décrit aussi comment nous avons validé par simulation l'apport de la solution proposée en termes de pouvoir de détection d'erreurs.

Chapitre 4 - Mise en œuvre de la fonction de contrôle d'erreur évolutive à base de codes CRC et validation par simulation

L'objectif de ce chapitre est double : 1) présenter tout d'abord une mise en œuvre de la fonction de contrôle évolutive que nous avons proposée dans le chapitre précédent, 2) décrire les travaux de validation effectués sur cette mise en œuvre.

Concernant la mise en œuvre, différents types de codes détecteurs pourraient être choisis au niveau des fonctions de contrôle composant la fonction de contrôle évolutive. Nous décrivons ici une mise en œuvre à base de codes CRC. Avant de présenter cette mise en œuvre, nous présentons d'abord les analyses préliminaires qui ont montré la pertinence de ce choix, et notamment la théorie mathématique sur laquelle il est possible de s'appuyer pour assurer un plus grand pouvoir de détection d'erreurs de la fonction de contrôle évolutive.

Une très large part de ce chapitre est ensuite consacrée à la validation par simulation de la mise en œuvre présentée précédemment, par rapport aux spécifications du cas d'étude, aussi bien en termes d'intégrité des communications que de respect des contraintes temporelles. Concernant l'intégrité, nous nous sommes focalisés sur l'intégrité vis-à-vis de l'altération des données (erreurs en valeur). Après avoir décrit l'environnement de simulation utilisé, nous présentons les modèles de simulation qui nous ont permis d'évaluer l'augmentation du pouvoir de détection d'erreurs de la fonction de contrôle évolutive, basée sur l'utilisation de codes CRC.

Finalement, pour valider la fonction de contrôle évolutive dans le cas de systèmes de CDV, nous vérifions si son apport en termes de pouvoir de détection d'erreurs permet de satisfaire l'objectif d'intégrité de haut niveau, à savoir un taux d'occurrence de l'événement redouté (embarquement des surfaces de contrôle) inférieur à 10^{-9}/h.

Concrètement, sur la base des résultats de simulation, nous montrons l'effet de l'application de la fonction de contrôle évolutive sur la diminution du taux d'embarquement des surfaces de contrôle, en présence de deux modes de défaillances des nœuds intermédiaires (collage de bits et erreurs d'adressage mémoire), et dont l'étude des risques présentée au chapitre 2 a montré qu'ils n'étaient pas couverts par les techniques classiques de détection d'erreurs.

4.1 Mise en œuvre de la fonction de contrôle évolutive

Avant d'étudier cette mise en œuvre, nous avons mené deux types d'analyses préliminaires que nous présentons maintenant.

4.1.1 Analyses préliminaires

La première analyse préliminaire concerne la détermination, pour une mise en œuvre logicielle, du temps de génération (côté émetteur) et de vérification (coté récepteur) des bits de CRC. La deuxième analyse a consisté à regarder sur quelles bases s'appuyer pour aboutir à un choix judicieux des différents polynômes générateurs à utiliser pour la fonction de contrôle évolutive.

4.1.1.1 Détermination du temps de génération/vérification des bits de CRC

Pour la mise en œuvre de la fonction de contrôle évolutive au niveau applicatif, nous avons envisagé d'utiliser des algorithmes logiciels pour la génération et la vérification des bits de CRC, comme pour la couche de sécurité dans [Brodtkorb 2001]. Le but de cette section est de vérifier si le temps d'exécution de ces algorithmes est compatible avec les contraintes temps réel des applications visées, notamment une période de 10 ms pour le rafraîchissement des surfaces de contrôle dans le cas des systèmes de CDV. Pour cela, le temps de génération et de vérification par logiciel doit être faible par rapport à la durée du temps de cycle (10 ms).

Rappelons que, dans une mise en œuvre matérielle des codes CRC, le temps de génération des bits de contrôle à l'émission d'un message et le temps de vérification de sa non altération n'ajoute aucun délai à la latence de sa transmission [Geremia 1999], contrairement à une mise en œuvre logicielle des codes CRC. Cela nous a amené à analyser les trois algorithmes standard suivants de mise en œuvre logicielle de codes CRC [Ramabadran & Gaitonde 1988].

- L'algorithme *Cyclic Redundancy Code Bitwise (CRCB)* qui reproduit logiciellement la mise en œuvre matérielle présentée au chapitre 1 (cf. figure 1.8), et basée sur des registres à décalage.
- L'algorithme *Cyclic Redundancy Code Table lookup (CRCT)* qui se base sur un tableau en mémoire contenant des valeurs pré-calculées représentant les restes des divisions polynomiales d'une série exhaustive de polynômes (de même degré) par le polynôme générateur du code CRC. Par exemple, pour un champ CRC de longueur 16, ce tableau contient 256 valeurs.
- L'algorithme *Cyclic Redundancy Code Reduced table lookup (CRCR)* qui se base sur un tableau réduit de valeurs pré-calculées représentant les restes de divisions polynomiales d'une série de polynômes particuliers (sous la forme x^i) par le polynôme générateur. Par exemple, pour un CRC 16 bits, ce tableau contient 8 valeurs.

Une présentation du fonctionnement de ces trois algorithmes, et de leurs codes assembleurs et un exemple de tableaux contenant des valeurs pré-calculées figurent dans l'annexe A.

À partir des codes assembleur de ces différents algorithmes [Schmidt 2000], et en se basant sur les caractéristiques d'un micro-contrôleur de type PIC18C542 [Microchip 2001] cadencé à une fréquence nominale de 20 MhZ, nous avons déterminé, pour des messages de 128 bits, le temps de génération et de vérification des bits de contrôle d'un code CRC 16 bits. Nous avons également déterminé la taille mémoire que nécessite l'exécution de chacun des trois algorithmes.

Les résultats de ces évaluations sont fournis par la figure 4.1. Le type de micro-contrôleur retenu pour cette évaluation (PIC18C542) est représentatif de ceux utilisés au niveau de l'Airbus A320. Les valeurs sont donc pessimistes par rapport aux valeurs que l'on pourrait obtenir avec des micro-contrôleurs susceptibles d'être embarqués dans le futur.

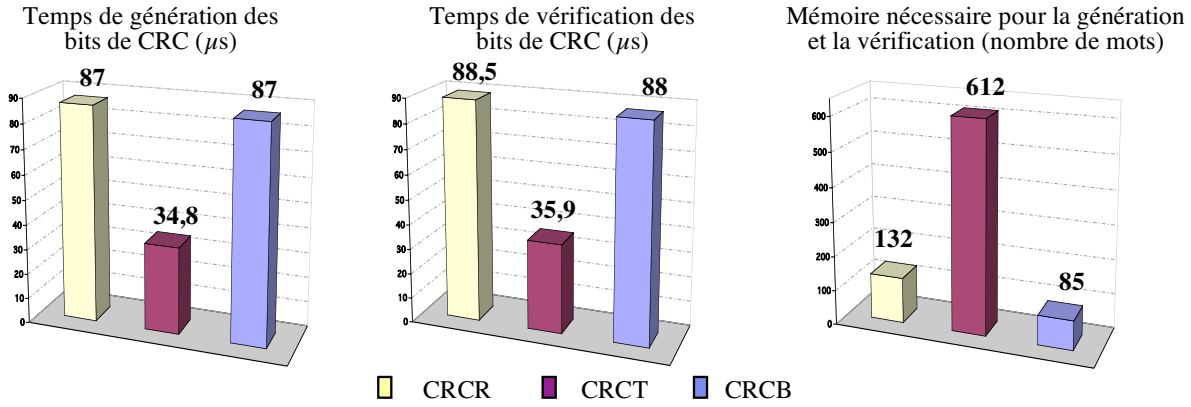


Figure 4.1 - Performances en termes de temps de calcul et de mémoire requise des algorithmes CRCR, CRCT et CRCB

Nous remarquons que l'algorithme CRCR est moins avantageux que l'algorithme CRCB, vu qu'il nécessite davantage de mémoire (132 mots pour CRCR et 85 pour CRCB), alors que les temps de génération et de vérification des bits de CRC sont presque identiques : 87 et 88,5 μs pour CRCR, 87 et 88 μs pour CRCB. Les algorithmes CRCT et CRCB sont donc les plus appropriés, mais le choix entre eux dépendra des contraintes imposées par le système et son environnement. En effet, l'algorithme CRCT est 3 fois plus rapide, en termes de temps de calcul, que l'algorithme CRCB, mais il nécessite 14 fois plus de mémoire.

En ce qui concerne le système de communication pour les CDV, un temps d'exécution de la fonction de contrôle qui est inférieur à 100 μs est tout à fait satisfaisant, même avec des micro-contrôleurs anciens. Vu qu'il n'y a pas de contraintes particulières en ce qui concerne la taille de la mémoire que nécessite la fonction de contrôle d'erreur, l'algorithme CRCT nous semble le plus approprié car il offre le temps d'exécution le plus faible.

4.1.1.2 Complémentarité des pouvoirs de détection d'erreurs

Étant donné que la fonction de contrôle évolutive se base sur un changement cyclique entre plusieurs fonctions de contrôle, il faut trouver un moyen pour que les différentes fonctions de contrôle utilisées soient les plus complémentaires possibles. Dans le cas de l'utilisation des codes CRC, le concept de complémentarité s'appuie sur les notions de polynômes primitifs [Peterson & Weldon 1972, p. 161], et sur la décomposition de polynômes générateurs en un produit de polynômes primitifs. Un polynôme est dit primitif, si et seulement si, le **pgcd** (plus grand commun diviseur) de ses coefficients est 1. On peut faire l'analogie avec un nombre premier qui n'est divisible que par 1 et lui-même.

Tous les mots d'un code CRC générés par un polynôme générateur en sont forcément tous multiples, et en contiennent donc un ou des facteurs primitifs. Puisque les erreurs non détectées par un code CRC sont nécessairement des multiples de son polynôme générateur, alors, pour avoir la probabilité maximale de détecter ces erreurs en utilisant un autre polynôme générateur, le produit des polynômes primitifs communs à ces deux polynômes doit être de plus petit degré possible. C'est ce concept de complémentarité qui guide le choix des polynômes générateurs utilisés par la fonction de contrôle évolutive, comme l'illustre la description qui suit.

4.1.2 Description de la mise en œuvre à base de codes CRC

Avec des codes CRC, la fonction de contrôle d'erreur évolutive consiste à changer (cycliquement par exemple) de polynôme générateur $G_1, G_2, \ldots, G_m$, d'un message à l'autre.

L'apport en termes de pouvoir de détection d'erreurs de la fonction de contrôle évolutive dépend du choix de ces polynômes générateurs. En effet, pour garantir un apport de détection optimal, les codes CRC générés par chaque polynôme générateur doivent avoir des pouvoirs de détection cumulables.

Nous allons présenter les caractéristiques des polynômes générateurs formant la fonction de contrôle évolutive, tout en décrivant la corrélation qui existe entre la complémentarité de leurs pouvoirs intrinsèques de détection des erreurs et le pouvoir de détection des erreurs de la fonction de contrôle évolutive qu'ils composent. Puis, nous présentons la technique permettant de choisir, ou de construire, ces polynômes.

4.1.2.1 Caractéristiques des polynômes générateurs

Les polynômes générateurs sur lesquels se base la fonction de contrôle évolutive devront avoir, d'une part, un pouvoir de détection d'erreurs important sur chaque message, et d'autre part, une complémentarité entre les pouvoirs de détection d'erreurs sur un lot de messages.

Si la condition de la complémentarité entre les pouvoirs de détection d'erreurs assurés par les m polynômes générateurs $G_1, G_2, \ldots, G_m$ est remplie, alors le pouvoir de détection de la fonction de contrôle évolutive représentant la série des m fonctions de contrôle $F_1, F_2, \ldots, F_m$ (basées sur ces polynômes respectifs) vis-à-vis d'un message erroné parmi N messages $M_1, M_2, \ldots, M_N$ consécutifs codés successivement par $F_1, F_2, \ldots, F_m, F_1, F_2, \ldots$ sera le même que celui d'une unique fonction de contrôle générant $\boldsymbol{m*p}$ bits (p étant le nombre de bits de contrôle) et appliquée à chacun des messages $M_1, \ldots, M_N$.

Certes, par rapport à l'utilisation d'une telle unique fonction de contrôle, une altération ne sera peut être détectée que par la fonction de contrôle F_i au message M_i, au lieu d'être détectée dès le premier message M_1. Mais en contre partie, moins de bits de contrôle auront été nécessaires.

Rappelons que le fait qu'une erreur ne soit détectée qu'au bout de i messages est compatible avec nos objectifs dans la mesure où on ne cherche pas à détecter les erreurs dans chaque message, mais dans un lot de messages.

La complémentarité entre les pouvoirs de détection d'erreurs des m fonctions de contrôle utilisées est d'autant plus importante que les m polynômes générateurs sur lesquelles elles se basent sont premiers entre eux. Il faut donc que ces polynômes aient en commun le plus petit facteur. Ce dernier, noté P, représente donc le polynôme de plus faible degré et qui permet de satisfaire un ensemble de propriétés en termes de détection d'erreurs, permettant à chacun des polynômes générateurs utilisés d'assurer un pouvoir de détection important ou "standard" sur chaque message, et notamment la détection à 100 % de certains types d'erreur.

Pour cela, le polynôme P devra satisfaire les propriétés génériques de la plupart des codes CRC standard pour la détection de certains types d'erreur. Ainsi, selon [Castagnoli *et al.* 1990], il est possible de garantir la détection de toutes les erreurs :

- simples, si le coefficient de x^0 de $G(x)$ est égal à 1,
- doubles, si $G(x)$ contient un polynôme primitif avec au maximum 3 termes,
- impaires, si $G(x)$ contient le facteur $1+x$.

Notons aussi à ce niveau, sans considérations particulières sur les polynômes générateurs, que les codes CRC permettent, par construction, de détecter toutes les erreurs :

- en rafale (*burst errors*), de longueur strictement inférieure à la distance de Hamming *d* du code,
- multiples, dont la multiplicité est strictement inférieure au degré *q* de *G(x)*.

Ainsi, pour garantir la détection des trois premiers types d'erreur, et pour que *P* ait le degré le plus faible, *P* devra être égal au produit des deux polynômes primitifs $1+x$ et $1+x+x^2$, soit $1+x^3$.

Dans ce qui suit, nous décrivons la technique utilisée pour aboutir à des polynômes générateurs ayant les caractéristiques décrites dans cette section, c'est-à-dire un pouvoir de détection standard pour chaque polynôme et des pouvoirs de détection complémentaires entre eux.

4.1.2.2 Choix des polynômes générateurs

Les critères de choix des différents polynômes générateurs entrant dans la constitution de la fonction de contrôle évolutive sont illustrés au travers d'un exemple sur la figure 4.2. On prend comme premier polynôme générateur, un polynôme noté G_1, décomposable en quatre polynômes primitifs P, P_1, P_2 et P_3 de degrés différents, avec P satisfaisant les propriétés génériques définies dans la section précédente. Donc, P entrera aussi dans la composition des différents polynômes générateurs utilisés par la fonction de contrôle évolutive.

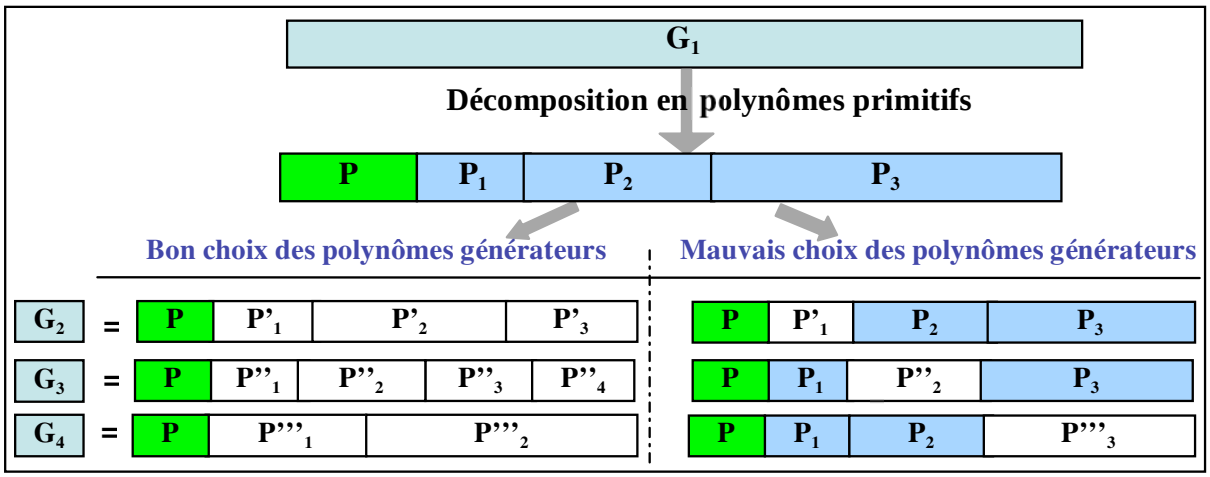


Figure 4.2 - Bon et mauvais choix de polynômes générateurs

Des polynômes générateurs choisis judicieusement sont des polynômes dont la composition est le produit du polynôme *P* et de un ou plusieurs polynômes primitifs différents des autres polynômes qui composent G_1, comme par exemple P'_1, P'_2 et P'_3 pour G_2. Dans ce cas, les polynômes générateurs ont, d'une part, un pouvoir de détection "standard" sur chaque message puisqu'ils contiennent le polynôme *P*, et d'autre part, ils ont le degré de complémentarité le plus élevé possible entre eux, en n'ayant que le polynôme *P* en commun.

A l'opposé, un mauvais choix de polynômes générateurs correspond à un produit incluant un ou plusieurs polynômes primitifs composant le polynôme générateur G_1. Ceci aura pour conséquence de diminuer la complémentarité au niveau de la détection des erreurs par les polynômes générateurs, et donc de diminuer l'effet de leurs pouvoirs cumulables de détection d'erreurs. L'apport de la solution proposée en termes de pouvoir de détection sera par conséquent plus faible.

4.2 Environnement de modélisation et de simulation

L'objectif principal des simulations est d'évaluer le gain procuré par l'utilisation de la fonction de contrôle évolutive, par rapport à un CRC standard, en termes de détection de messages erronés, et de vérifier si l'objectif d'intégrité fixé au niveau du système de communication est atteint : ne pas avoir plus de *X* messages erronés non détectés dans un lot de *N* messages. Rappelons que, dans le cas des systèmes de CDV, pour *X* et *N* fixés respectivement à 3 et 10, atteindre cet objectif permet de valider l'objectif d'intégrité de haut niveau, qui est un taux d'embarquement d'une surface de contrôle inférieur à 10^{-9}/h.

Après avoir présenté l'outil de simulation retenu (Matlab-Simulink), nous décrivons ensuite certains modules de base de l'outil et les modules que nous avons nous-mêmes développés, et sur lesquels nous nous sommes appuyés pour développer les modèles destinés à la validation de la fonction de contrôle évolutive.

4.2.1 Outil de simulation retenu : Matlab-Simulink

Comme outil de modélisation/simulation adapté à nos besoins, nous avons recherché un outil :

1) capable de simuler une chaîne complète de communication : sources et récepteurs de données, canal de transmission, génération et vérification des bits de contrôle permettant de mettre en œuvre facilement le codage CRC ;
2) capable de simuler la couche physique d'un système de communication, c'est-à-dire pouvoir manipuler des bits et des trames ;
3) ouvert, afin de pouvoir ajouter des modules utilisateurs qui n'existent pas par défaut dans l'outil de base.

Après une analyse de plusieurs outils de simulation, dont en particulier Comnet, Opnet, VisSim, SystemView, notre choix s'est porté sur l'outil **Matlab-Simulink**.

Matlab est une plateforme de calcul scientifique, distribuée par la société *The MathWorks* (www.mathworks.com). En plus de ses capacités de calcul et d'analyse, cette plateforme offre tout un ensemble de fonctionnalités graphiques permettant de visualiser les résultats. Matlab est constitué d'un noyau relativement réduit, capable d'interpréter, puis d'évaluer les expressions numériques matricielles qui lui sont adressées (cf. figure 4.3) :

- soit directement au clavier, depuis une fenêtre de commande ;
- soit sous forme de séquences d'expressions (ou *scripts*) enregistrées dans des fichiers texte appelés *m-files,* et exécutées depuis la fenêtre de commande ;
- soit, plus rarement, sous forme de fichiers binaires appelés *mex-files,* générés à partir d'un compilateur C ou Fortran.

Ce noyau est complété par une bibliothèque de fonctions prédéfinies, très souvent sous forme de fichiers *m-files*, et regroupés en paquetages ou *toolboxes*. À côté des *toolboxes* requises *local* et *matlab*, il est possible d'ajouter des toolboxes spécifiques à des domaines particuliers, dont notamment la *Communications toolbox* et la *Signal Processing toolbox* qui offrent des fonctions pour modéliser la couche physique d'un système de communication.

Simulink, quant à lui, est un outil graphique interactif de modélisation, de simulation et d'analyse de systèmes dynamiques. Il représente l'extension graphique de Matlab permettant de construire des schémas bloc, de simuler le comportement d'un système, d'évaluer ses performances et d'affiner la conception. Simulink s'intègre de manière transparente au dessus de Matlab et permet la conception de systèmes de contrôle, de communication, etc.

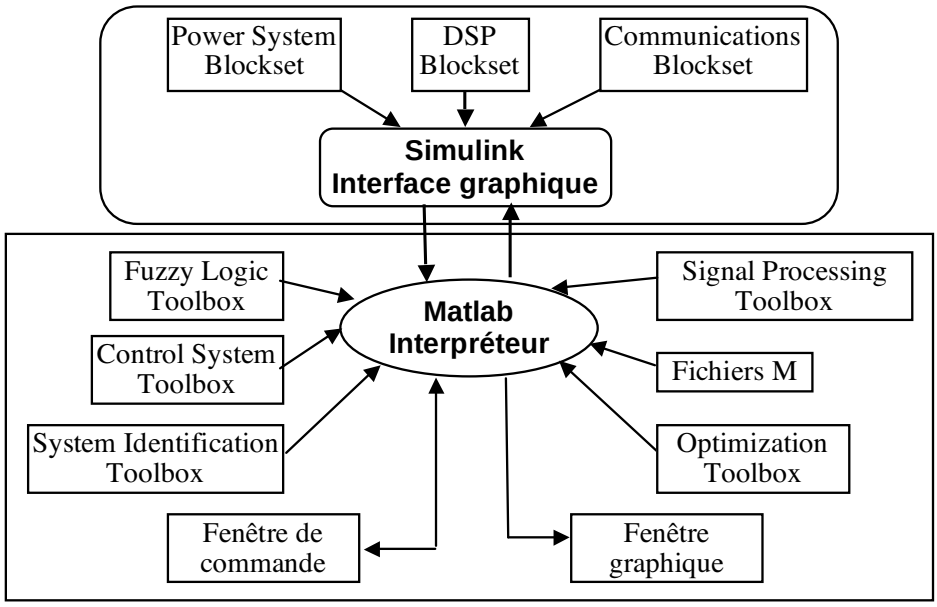


Figure 4.3 - Environnement Matlab

Simulink s'appuie également sur un ensemble de bibliothèques de blocs applicatifs, dénommés *blocksets* (cf. figure 4.3), couvrant de multiples domaines (modélisation de systèmes électriques de puissance, traitement numérique du signal, etc.).

L'intérêt fort pour nous est l'existence en particulier du *Communications blockset* qui contient plus de 150 blocs pour modéliser la couche physique d'un système de communication, et en particulier l'existence (dans la Release 13) d'un bloc générique pour simuler le codage CRC, avec la possibilité pour l'utilisateur de spécifier ses propres polynômes générateurs.

Notons aussi la possibilité de générer directement du code en langage C ou ADA à partir des modèles Simulink, ceci pour accélérer les simulations ou générer des applications embarquées. Enfin, il est possible de développer des blocs personnalisables pouvant incorporer des codes Matlab, C, Fortran et Ada existants (blocs *S-functions*).

Décrivons maintenant les modules de base de l'outil Matlab-Simulink qui cadrent avec nos besoins de simulation.

4.2.2 Modules de base

Nous fournissons ici une description des fonctions et des possibilités offertes par certains blocs de base de l'outil Matlab-Simulink pour simuler différents éléments d'une chaîne de communication, et qui ont servi à l'élaboration des modèles présentés par la suite. Il s'agit des sources de données, du codage et du décodage canal CRC.

4.2.2.1 Sources de données

Les sources de données sont modélisables par le bloc *Bernoulli Binary Generator* (du blockset *Communications*) qui génère une suite binaire aléatoire suivant une distribution de Bernoulli. Une distribution de Bernoulli avec un paramètre p génère un *0* avec une probabilité p et génère un *1* avec une probabilité de $1-p$.

4.2.2.2 Codage canal – CRC

Il existe des blocs de base génériques de Simulink pour modéliser un codage CRC : les blocs *CRC Generator* et *CRC-N Generator* (du blockset *Communications*).

Le bloc *CRC Generator* génère un checksum pour chaque donnée en entrée et le place en fin de message. Le polynôme générateur est spécifié comme paramètre sous forme d'un vecteur binaire représentant les coefficients du polynôme, ou sous forme d'un vecteur d'entiers contenant les puissances des termes non nuls du polynôme. Il est possible aussi de spécifier le nombre de checksums par trame en paramètre. Une version simplifiée de ce bloc est disponible avec le bloc *CRC-N Generator*, où N représente le degré du polynôme générateur. Ce dernier bloc permet de choisir le polynôme générateur parmi une liste limitée de polynômes couramment utilisés, présentés dans le tableau 4.1.

Code CRC	***Polynôme générateur***	***Nombre de bits***
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	32
CRC-24	$x^{24}+x^{23}+x^{14}+x^{12}+x^8+1$	24
CRC-16	$x^{16}+x^{15}+x^2+1$	16
Reversed CRC-16	$x^{16}+x^{14}+x+1$	16
CRC-8	$x^8+x^7+x^6+x^4+x^2+1$	8
CRC-4	$x^4+x^3+x^2+x+1$	4

Tableau 4.1 - Les polynômes générateurs offerts par le bloc *CRC-N Generator*

4.2.2.3 Décodage canal

Le décodage canal se fait à l'aide des blocs *General CRC Syndrome Detector* et *CRC-N Syndrome Detector* (du blockset *Communications*). Le bloc *General CRC Syndrome Detector* reçoit le message et en retire le checksum reçu. Il calcule ensuite le checksum et le compare avec celui reçu. Ce bloc a deux sorties. La première fournit le message reçu sans le checksum et la deuxième (variable *CRC_syndrome*) renvoie une valeur booléenne égale à : 0 si les deux checksums correspondent, et sinon 1. Les paramètres de ce bloc doivent être cohérents avec ceux du bloc codeur *CRC Generator*. Le bloc *CRC-N Syndrome Detector* est une version simplifiée du bloc précédent avec des polynômes générateurs prédéfinis. Il est le bloc dual du bloc codeur *CRC-N Generator* de l'émetteur.

4.2.3 Modules développés

Les modules supplémentaires que nous avons développés concernent le multiplexage et démultiplexage canal, l'introduction des différents types d'erreur de transmission identifiés au chapitre 2, et enfin le calcul des taux d'erreur.

4.2.3.1 Multiplexage-démultiplexage canal (transmission)

Dans le cadre de notre cas d'étude, les transmissions sur un canal de communication utilisent un multiplexage-démultiplexage de type temporel (TDMA). Deux méthodes sont possibles en faisant appel à la combinaison de plusieurs blocs.

a) Première méthode

La phase de multiplexage TDMA peut être modélisée à l'aide du bloc *Multiport Switch* (de la bibliothèque *signal routing* de Simulink), qui permet de multiplexer ses signaux d'entrée, selon des paramètres reçus sur son premier port pour fixer lequel de ces signaux est à diriger vers sa sortie à un instant donné. La figure 4.4 montre un exemple de multiplexage de trois signaux, à l'aide du signal [1 2 3], qui permet de choisir dans l'ordre les trois entrées pendant un temps d'échantillonnage spécifié dans un masque défini dans le bloc *Signal From Workspace*.

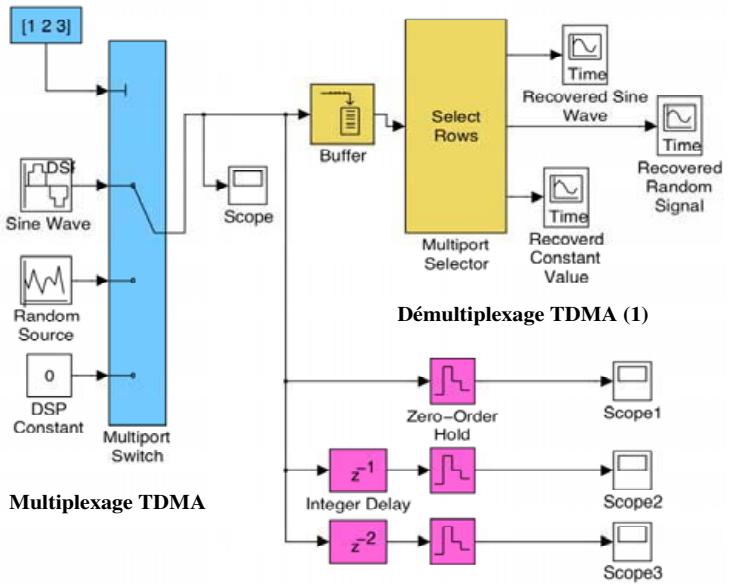


Figure 4.4 – Principe de la 1ère méthode de Multiplexage – Démultiplexage

Quant au **démultiplexage** TDMA, il se modélise de deux façons avec Simulink, illustrées sur l'exemple de la figure 4.4 :

1) TDMA(1) : une zone tampon collecte 300 échantillons où 100 d'entre eux appartiennent à chacun des trois signaux, puis le bloc *Multiport Selector* (du blockset *signal processing*) permet de démultiplexer les trois signaux comme suit. Les échantillons 1, 4, 7 sont envoyés sur la 1ère sortie, les échantillons 2, 5, 8 sur la 2ème sortie et les échantillons 3, 6, 9 sur la 3ème sortie.
2) TDMA(2) : on utilise un bloc *Zero-Order Hold Selector* (de la bibliothèque *discrete* de Simulink) pour récupérer le premier signal. Ce bloc échantillonne et maintient la valeur du signal pendant la durée spécifiée en paramètre et correspondant à la durée de l'échantillon (*sample time*) du signal émis. Pour récupérer le 2ème signal, on utilise un 2ème bloc *Zero-Order Hold*, mais précédé cette fois d'un bloc *Integer Delay* qui introduit un décalage de la durée d'un échantillon (héritée du bloc *Multiport Switch*). Pour récupérer le 3ème signal, on introduit un délai de deux échantillons.

b) Deuxième méthode

Le multiplexage peut aussi être modélisé à l'aide des blocs *Matrix Concatenation* et *Matrix Interleaver* de Simulink (de la bibliothèque *mathoperation* de Simulink). Ce dernier bloc permet de permuter les symboles en entrée en écrivant dans une matrice ligne par ligne, et en la lisant colonne par colonne. Le produit du nombre de lignes et du nombre de colonnes de la matrice spécifiée doit correspondre à la taille des données en entrée.

Le démultiplexage est modélisé avec le bloc *Multiport Selector* qui extrait des sous-ensembles de la matrice ou du vecteur en entrée, et les dirige vers le port adéquat. Chaque sous-ensemble est défini par un vecteur précisant les indices à extraire. La figure 4.5 illustre cette méthode.

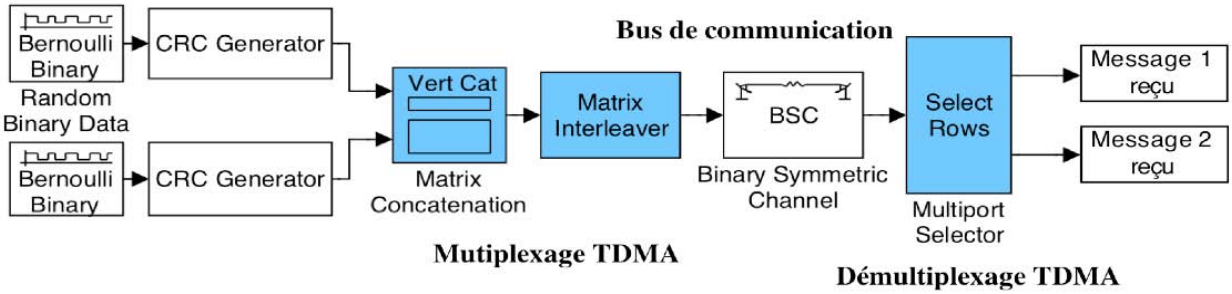


Figure 4.5 – Principe de la 2[d] méthode de Multiplexage – Démultiplexage

Par la suite, dans toutes les étapes de modélisation du système de communication envisagé pour les CDV, nous utilisons cette seconde méthode, à la fois plus simple, car elle utilise moins de blocs, et plus rapide, car elle n'ajoute pas de délai, les signaux étant sous forme de trames.

4.2.3.2 Modélisation des erreurs

Nous indiquons maintenant comment sont introduits dans nos simulations les différents modèles d'erreur qui ont été proposés (cf. chapitre 2) suivant les types de risques à couvrir.

Les *risques liés au bruit,* qui se manifestent par des inversions aléatoires de bits, sont modélisés à 2 niveaux : directement au niveau du canal en fixant un taux de BER, ou alors de manière plus fine à l'aide du bloc *Binary Vector Noise Generator Error* (du blockset *communications*). Ce bloc génère un vecteur bruit binaire dont la longueur est spécifiée en paramètre. La position et le nombre de 1 contenus dans ce vecteur représentent les inversions de bits. Le paramètre *Probabilities* permet de déterminer combien de 1 apparaissent dans le vecteur bruit. Leur emplacement suit une distribution uniforme. Si *p1*, *p2*, ..., *pm* est l'entrée du paramètre *Probabilities*, alors *p1* est la probabilité que le vecteur généré contienne un seul 1, *p2* est la probabilité d'avoir deux 1, etc., sachant que la somme des probabilités doit être inférieure ou égale à 1. Ce bloc est utile pour tester les codes détecteurs et correcteurs d'erreurs.

Les *risques liés aux défaillances de câblage,* qui se manifestent par des erreurs en rafale (ou burst errors), sont modélisés dans un script Matlab par un vecteur contenant un bloc de 1 apparaissant à des intervalles aléatoires. Par exemple, avec un code Matlab, on peut créer un vecteur d'erreur dans lequel des blocs de trois 1 apparaissent à des intervalles aléatoires. La distance entre deux blocs consécutifs de 1 est un entier aléatoire compris entre 1 et 80.

Les *risques liés aux défaillances des nœuds d'interconnexion,* qui se manifestent par un collage de bits, sont modélisés par un vecteur, créé par un code Matlab, ayant des bits à valeur fixe.

4.2.3.3 Calcul des taux d'erreur (bits, messages)

Le bloc *Error Rate calculation* (de la bibliothèque *Comm Sinks* de Simulink) permet de calculer le taux d'erreur, bit ou plus généralement symbole, des données en entrée. Si les données en entrée sont des bits, alors ce bloc calcule le taux d'erreur bit, si les données en entrée sont des symboles, alors il calcule le taux d'erreur symbole. Ce bloc reçoit en entrée les données *Tx* émises et les données *Rx* reçues. Il les compare et calcule le taux d'erreur en divisant le nombre total de bits différents par le nombre total de bits d'une seule entrée.

Pour les données en sortie de ce bloc, il s'agit d'un vecteur de trois éléments correspondant : 1) au taux d'erreur, 2) au nombre total d'erreurs correspondant au nombre total d'éléments différents (bits ou trames), et 3) au nombre total de comparaisons effectuées.

Ce bloc est très important vu qu'il permet, d'une part, l'arrêt des simulations en se basant sur les statistiques d'erreurs, et d'autre part, le calcul du taux de non détection des messages erronés. Sur la base de la valeur de ce taux, se fera par la suite la validation de la technique que nous avons défini, par rapport aux objectifs d'intégrité fixés.

a) Arrêt de la simulation en se basant sur les statistiques d'erreurs

Pour toute simulation, se pose la question du point d'arrêt : quand et sur quoi ? Dans notre contexte, une manière de procéder est d'utiliser le bloc *Error Rate Calculation*, qui peut être configuré afin que les statistiques d'erreurs contrôlent la durée de la simulation. Par défaut, la simulation est arrêtée lorsque un nombre maximal d'erreurs, défini par l'utilisateur (*Target number of errors*) est atteint. Mais elle peut aussi être arrêtée en fixant : 1) un nombre maximal de symboles transmis (spécifié en paramètre du bloc *Error Rate Calculation*), ou 2) une durée à l'aide du paramètre *Stop time* (dans la boîte de dialogue *Simulation parameters*).

Pour ignorer le ou les deux derniers critères d'arrêt, il faut affecter aux paramètres correspondants la valeur *In1*. Ceci, dans le but d'atteindre un certain nombre d'erreurs sans avoir à arrêter la simulation de façon prématurée. Notons qu'il ne s'agit pas là des seuls moyens d'arrêter la simulation, d'autres blocs peuvent conduire à l'arrêt.

b) Calcul du taux de non détection de messages erronés

Rappelons que notre objectif, en termes d'intégrité des transmissions, est d'assurer un taux de non détection de X messages erronés dans un lot de N messages inférieur à un seuil fixé (ex. : 3 messages sur 10). Le simulateur doit donc intégrer une fonction permettant de calculer ce taux. Or, contrairement aux fonctions décrites jusqu'ici, Matlab-Simulink ne propose pas par défaut une telle fonction. Nous avons donc été amené à construire, à partir des blocs de base de Simulink, des sous systèmes autour du bloc de base *Error Rate Calculation*. Ces systèmes sont détaillés dans ce qui suit, et illustrés dans la figure 4.6.

Précisons, que dans le cas d'un contrôle d'erreur par CRC, si un message erroné est non détecté, et est multiple du polynôme générateur, alors il produit au niveau du récepteur une valeur nulle de la variable booléenne *CRC_syndrome*. Dans le cas où ce syndrome est nul (ou u1 = 0 dans la figure 4.6), le bloc If_1 déclenchera l'exécution du bloc *If action* qui se base sur le bloc *Error Rate Calculation*. Ce dernier permet le calcul du taux de messages erronés non détectés en comparant les deux données en entrée : une constante égale à 0 et une variable entière résultante d'une opération logique XOR comparant bit à bit les messages émis et reçus.

La somme de toutes les comparaisons binaires permet de savoir si le message reçu est différent de celui émis (message erroné). Si le résultat est égal à 0, cela indique qu'il n'y a pas eu d'erreur détectée, sinon il y a au moins une erreur dans le message reçu.

Sur la base de ces comparaisons, le bloc *Error Rate Calculation* génère trois données :

1) le nombre de différences qui existent entre les deux messages en entrée (nombre de messages reçus et nombre de messages erronés),
2) le nombre total de comparaisons effectuées (nombre de messages reçus),
3) le rapport entre les deux données en entrée du bloc (taux de trames reçues et erronées).

Ce rapport traduit en fait le taux de trames erronées et non détectées, puisque le bloc *If Action* contenant le bloc *Error Rate Calculation* ne se déclenche que si le *CRC_syndrome* vaut 0.

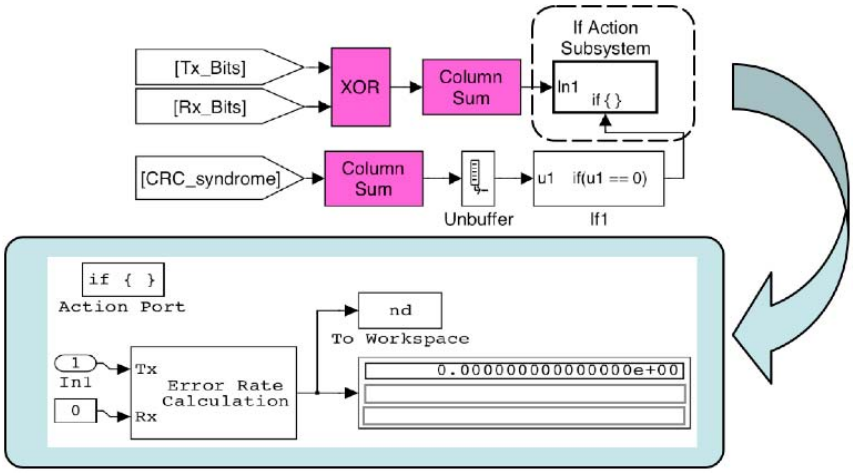


Figure 4.6 - Calcul du taux des messages erronés non détectés

Par ailleurs, nous présentons dans l'annexe B d' exemples de modélisations réalisées, en utilisant les blocs de base et les sous-systèmes de l'outil Matlab-Simulink, qui ont été introduits ci-avant.

4.3 Modèles de simulation pour la validation de la fonction de contrôle d'erreur évolutive

Ces modèles ont pour but la validation de la fonction de contrôle d'erreur évolutive, et plus précisément la validation du changement cyclique entre des polynômes générateurs. Ils doivent permettre d'analyser son apport en termes de pouvoir de détection des erreurs dans un lot de messages transmis de bout en bout du système de communication.

En prenant pour ces simulations le contexte des systèmes de CDV, nous pouvons valider les résultats du chapitre précédent, qui ont permis de déterminer le nombre minimal de fonctions de contrôle permettant de ramener le taux d'embarquement d'une surface de contrôle à une valeur inférieure au seuil de criticité de10^{-9}/heure de vol.

En effet, nous avons montré que cet objectif pouvait être atteint en utilisant seulement deux fonctions de contrôle ayant des pouvoirs de détection complémentaires, et sous des conditions particulières de défaillance des nœuds intermédiaires et de stratégie de recouvrement des surfaces de contrôle. Reste donc à vérifier ces résultats par simulation, pour valider l'apport de la fonction de contrôle évolutive.

4.3.1 Description des modèles de simulation

Rappelons que l'idée de base de la fonction de contrôle évolutive est de changer cycliquement le polynôme générateur qui génère les bits de contrôle de CRC, afin qu'une erreur répétitive et non détectée par l'un des polynômes soit détectée par au moins l'un des autres.

Ce principe du changement cyclique entre plusieurs polynômes générateurs est mis en œuvre dans nos modèles avec le bloc *Multiport Switch*, qui va diriger de façon cyclique une de ses entrées (correspondant chacune à un message codé avec l'un des polynômes) vers la sortie selon la valeur du port de contrôle.

Côté récepteur, nous utilisons également le bloc *Multiport Switch* avec les mêmes paramètres afin que le polynôme générateur utilisé pour le décodage soit le même que celui utilisé pour le codage.

Un autre point important de notre simulateur est que, pour réduire de manière très significative les temps de simulation, nous injectons (transmettons) uniquement des erreurs non détectables par l'un des polynômes, et nous déterminons par simulation le taux de non détection de ces erreurs par les autres polynômes. L'erreur injectée dans l'outil de simulation est générée par l'un des polynômes générateurs utilisés, et donc elle en est forcément multiple. Dans tous les exemples ci-après, la génération de ces erreurs se fera à partir du premier polynôme générateur du cycle, noté G_1.

Deux formes de génération des erreurs ont été testées : une génération aléatoire et une génération exhaustive. Ce qui conduit aux deux types de modèles présentés ci-après. Nous ferons également plus loin dans ce chapitre une analyse comparative entre ces deux modes de génération d'erreurs.

4.3.1.1 Modèle avec génération aléatoire des erreurs

Concrètement, une erreur est introduite via un produit de convolution (modulo 2) de la forme binaire du polynôme générateur et d'une série de bits générée aléatoirement suivant la loi de Bernoulli. La figure 4.7 présente le détail du modèle réalisé.

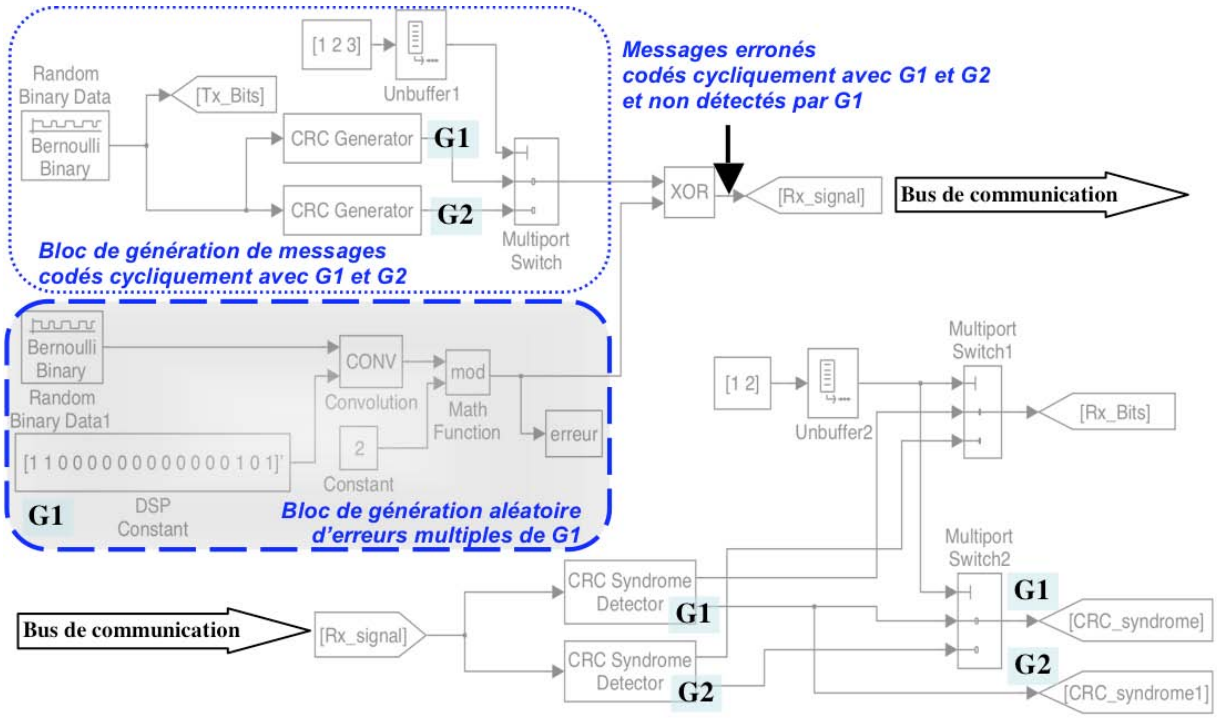


Figure 4.7 - Injection ALEATOIRE d'erreurs multiples de G_1 pour un changement cyclique avec deux polynômes générateurs (G_1 et G_2)

4.3.1.2 Modèle avec génération exhaustive des erreurs

Pour la génération **exhaustive** de messages erronés multiples du polynôme générateur G_1, les erreurs sont également introduites via un produit de convolution (modulo 2) de la forme binaire du polynôme générateur et d'une série de bits, mais celle-ci est cette fois générée de façon exhaustive avec un compteur binaire, comme le montre la figure 4.8, qui n'explicite que la partie du modèle de la figure 4.7 (partie grisée) qui change, les autres parties étant réduites aux blocs hachurés.

Cependant, notre génération exhaustive des erreurs est particulière, dans le sens qu'elle ne correspond qu'à une partie des messages (les 32 bits de poids le plus faible), qui devraient théoriquement être générés. En effet, nous générons, avec un compteur allant de 0 à 2^{32}-1, un vecteur de 32 bits correspondant à la représentation binaire de la valeur décimale du compteur. Ce vecteur est convolué (modulo 2) avec le vecteur de 16 bits correspondant au polynôme générateur G_1, ce qui conduit à un vecteur de 47 bits. À ce produit de convolution, nous ajoutons une série de 69 bits à 0 formant ainsi un message erroné de 116 bits (la taille des messages codés étant prise égale à 116), qui est multiple de G_1 et donc non détectable. Ces particularités sont dues aux limitations des blocs de base du simulateur (*Counter* et *Integer to bit Converter*), et à la durée considérable de simulation que peut prendre le processus de génération exhaustive d'une série de 2^{116}-1 vecteurs de 116 bits. D'où la limitation à un compteur allant jusqu'à 2^{32}-1.

En répétant le processus décrit ci-dessus à chaque nouvelle incrémentation du compteur, nous générons de façon exhaustive une série de messages erronés, codés de façon alternée avec les polynômes générateurs G_1 et $G2$, et qui sont non détectables par G_1.

Cette génération exhaustive des erreurs est moins réaliste que la génération aléatoire, puisque la forme particulière des messages erronés générés se composent de blocs de 69 bits à 0.

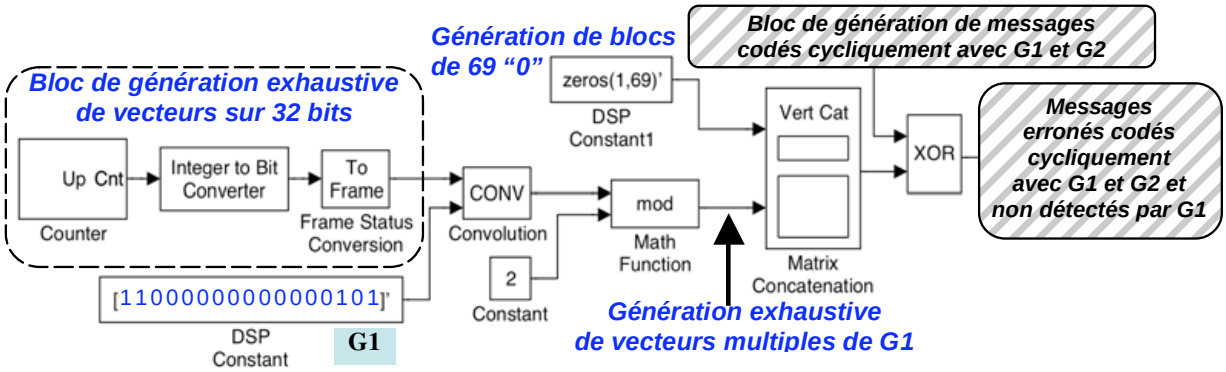
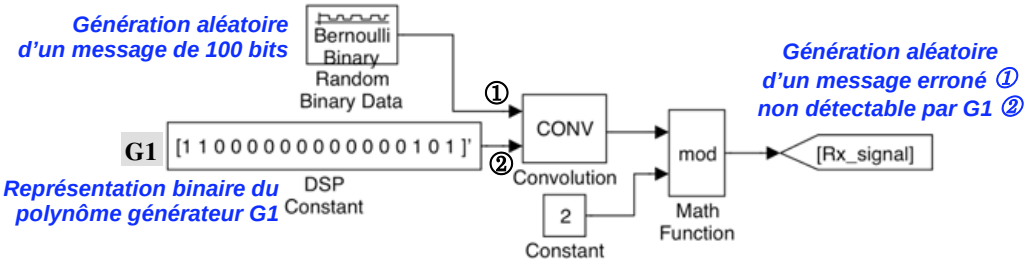


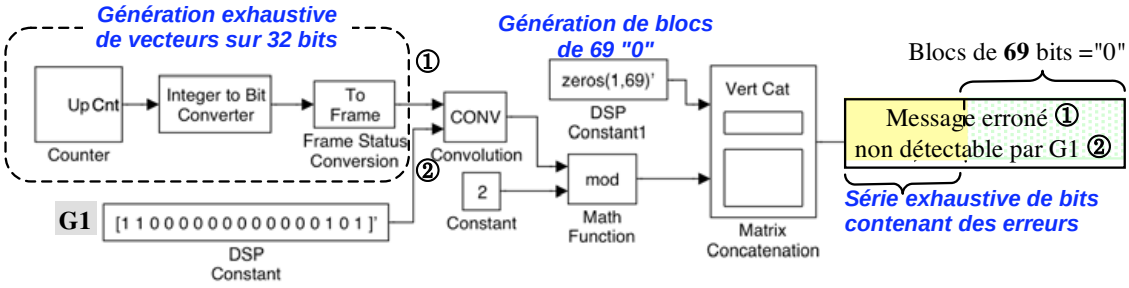
Figure 4.8 - Injection EXHAUSTIVE d'erreurs multiples de G_1 pour un changement cyclique avec deux polynômes générateurs (G_1 et G_2)

4.3.2 Simplification des modèles de simulation

Pour réduire de façon encore plus significative le temps d'exécution des simulations, nous allons simplifier encore plus le processus de génération et d'injection des erreurs. Pour cela, au lieu de coder cycliquement (avec des polynômes générateurs) les messages générés (données utiles) avant de les convoluer avec des vecteurs d'erreurs multiples du polynôme G_1, nous transmettons directement des messages supposés à l'avance codés et erronés et non détectables par G_1. Ces messages sont obtenus en convoluant une série de bits, générée aléatoirement ou exhaustivement, avec la forme binaire de G_1 (cf. figures 4.9a et 4.9b).



4.9a - Génération aléatoire de messages erronés non détectables par G1



4.9b - Génération exhaustive de messages erronés non détectables par G1

Figure 4.9 - Modèles simplifiés d'injection d'erreurs

Dans ces nouveaux modèles, et avec un changement cyclique basé sur deux polynômes générateurs G_1 et G_2, nous ne considérons donc plus le cas où les messages sont codés une fois sur deux avec G_2, avant d'être convolués avec le vecteur d'erreur multiple de G_1. Les messages erronés ainsi formés ne sont pas forcément des multiples de G_1, et peuvent donc aussi être détectés par ce dernier.

En effet, le message erroné transmis dans ces nouveaux modèles est systématiquement un multiple de G_1, alors que dans l'ancien modèle, c'est le vecteur d'erreur généré qui est systématiquement un multiple de G_1 ($A.G_1$). Et puisque ce dernier est une fois sur deux convolué avec un message codé avec G_2 ($B.G_2$), alors le message erroné ($A.G_1+B.G_2$) ainsi formé n'est pas forcément un multiple de G_1 et peut donc être détecté comme erroné.

Pour le calcul du taux de messages erronés non détectés, nous utilisons la technique proposée à la section 4.2.3.3.b, mais en supprimant cette fois la partie qui se base sur des comparaisons entre les messages émis et reçus, puisqu'on n'émet plus que des messages erronés. Donc, la technique de calcul du taux d'erreur non détectées ne repose plus que sur la valeur booléenne du bloc *CRC_Syndrome* : une valeur nulle correspond forcément à la non détection du message erroné par un des polynômes générateurs utilisés, et une valeur égale à 1 correspond à la détection du message erroné.

Ces derniers modèles de simulation sont adaptés par la suite à plus de 2 polynômes générateurs. C'est sur la base de ces nouveaux modèles qu'ont été effectuées toutes les expérimentations, ainsi que l'analyse et l'interprétation des résultats qui en sont tirés.

4.3.3 Généralisation des modèles de simulation à "*m*" polynômes générateurs

Ces modèles généralisés présentent donc un changement cyclique sur *m* polynômes générateurs, et la modification apportée par rapport aux modèles précédents correspond à un paramétrage du bloc *Multiport switch* en spécifiant "*m*" entrées au lieu de 2, et induit donc l'ajout de "2.*m*" blocs supplémentaires pour le codage et le décodage des champs CRC générés par les différents polynômes générateurs utilisés (cf. figure 4.10).

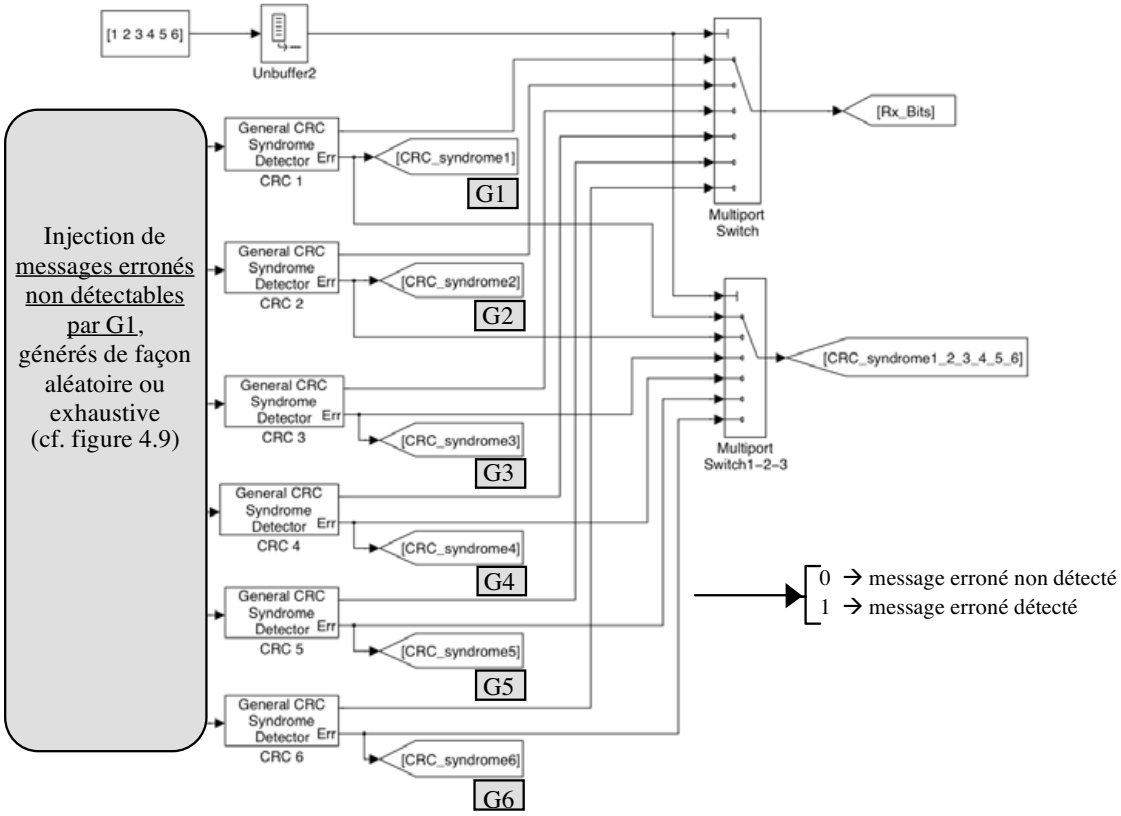


Figure 4.10 - Modèle simplifié de simulation du changement cyclique à six polynômes générateurs

Les différents polynômes générateurs sont choisis selon les caractéristiques de leurs pouvoirs de détection d'erreurs, à la fois intrinsèques et combinés. Des simulations ont été réalisées avec un "bon" et un "mauvais" choix de polynômes générateurs, dans le but de montrer la corrélation existant entre l'apport, en termes de pouvoir de détection d'erreurs, du changement cyclique des polynômes générateurs et leurs pouvoirs intrinsèques de détection d'erreurs.

En outre, certains des polynômes générateurs utilisés dans les modèles de simulation sont des polynômes standard 16 bits (les plus utilisés dans l'industrie), et nous montrons, par simulation, que l'apport des changements cycliques avec des polynômes générateurs standard peut être dans ce cas optimal pour certaines combinaisons de ces polynômes.

Il est important de préciser que ces simulations valident aussi la couverture, via le changement cyclique de polynômes générateurs, des risques liés aux défaillances fonctionnelles des tampons de stockage des nœuds intermédiaires. À partir du modèle de la figure 4.10, nous utilisons les différentes valeurs booléennes des blocs *CRC_syndrome* pour calculer, dans le modèle présenté à la figure 4.11, le taux de non détection des messages erronés (initialement égal à 1) en utilisant seulement le polynôme G_1, par la fonction de contrôle d'erreur évolutive.

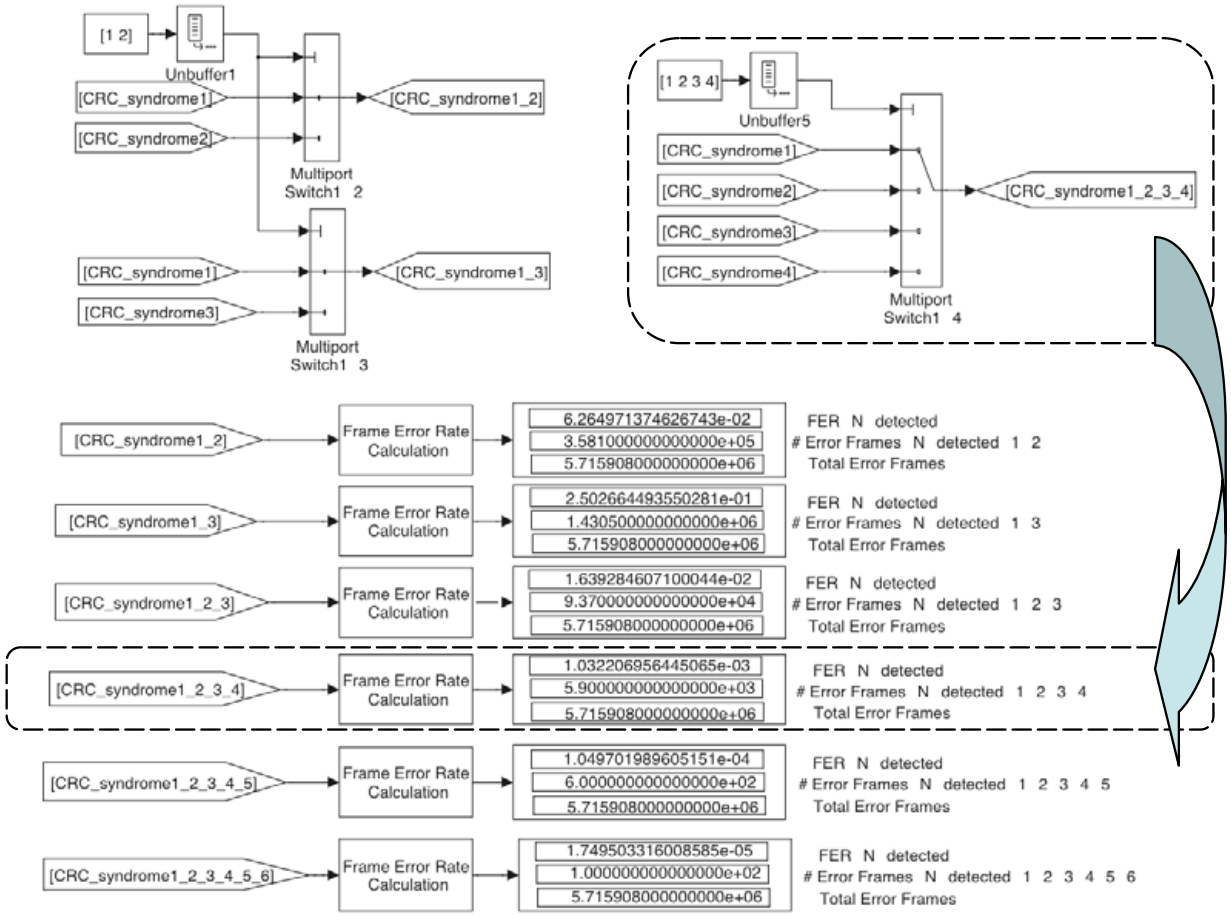


Figure 4.11 - Modèle de calcul du taux de non détection des messages erronés par le changement cyclique entre des polynômes générateurs

Dans cet exemple, nous cherchons à calculer le taux de non détection des messages erronés par le changement cyclique entre m polynômes générateurs ($m \in [3, 6]$) : G_1, G_2 et … G_m. Pour cela, nous transmettons de façon alternée les valeurs booléennes des blocs *CRC_syndrome_i* ($i \in [1, m]$) au bloc *CRC_syndrome 1_2_…_m*.

Le nombre de toutes les valeurs que contient ce dernier bloc correspond au nombre de messages reçus et qui, à la base, sont tous erronés et multiples de G_1. Le nombre de valeurs à 1 correspond au nombre de messages erronés et non détectés par un des m polynômes générateurs. Nous utilisons alors un bloc *Frame Error Rate Calculation* en le paramétrant de sorte qu'il compare avec la valeur 0 toutes ses entrées, correspondantes aux sorties du bloc *CRC_syndrome 1_2_…_m*. Le nombre de différences entre 0 et les valeurs en entrée donne le nombre de messages erronés et non détectés par l'un des m polynômes générateurs utilisés.

Par la suite, sur la base des modèles de simulation des figures 4.10 et 4.11, nous évaluons l'apport de la fonction de contrôle évolutive en termes de pouvoir de détection d'erreurs.

Une première étape consiste à choisir les polynômes générateurs et à évaluer leurs pouvoirs intrinsèques de détection d'erreurs. Nous choisissons également, suite à une interprétation des pouvoirs de détection des erreurs générées aléatoirement et de façon exhaustive, le type de génération correspondant le mieux à nos besoins de validation.

4.4 Évaluation du pouvoir de détection de la fonction de contrôle évolutive et validation de son apport

Nous évaluons le pouvoir de détection d'erreurs de la fonction de contrôle évolutive en se basant sur un bon et un mauvais choix de polynômes générateurs. Ces choix reposent sur la complémentarité des polynômes générateurs utilisés avec le polynôme G_1 et entre eux.

Le tableau 4.2 ci-dessous présente un exemple de choix des polynômes générateurs. Les polynômes $G_2(x)$, $G_4(x)$ et $G_5(x)$ représentent un bon choix vu qu'ils n'ont en commun avec G_1 que le facteur $1+x$. Par contre les polynômes $G_3(x)$ et $G_6(x)$ représentent un mauvais choix vu qu'ils ont, en plus de $1+x$, le facteur $1+x+x^7$ en commun avec $G_1(x)$.

<table>
<tr><td colspan="5">$G_1(x) = \underline{(1+x)}.\ \underline{(1+x+x^7)}\ .\ (1+x^2+x^3+x^4+x^8)$</td></tr>
<tr><td colspan="3">Bon choix de polynômes générateurs</td><td colspan="2">Mauvais choix de polynômes générateurs</td></tr>
<tr><td colspan="3">$G_i(x)$ = (1+x) * un polynôme primitif de degré 7
* un polynôme primitif de degré 8</td><td colspan="2">$G_i(x) = (1+x)*\ (1+x+x^7)$
* un polynôme primitif de degré 8</td></tr>
<tr><td rowspan="2">Identifiant</td><td colspan="2">Forme</td><td rowspan="2">Identifiant</td><td colspan="2">Forme</td></tr>
<tr><td>développé</td><td>décomposée</td><td>développé</td><td>décomposée</td></tr>
<tr><td>$G_2(x)$</td><td>$1+x+x^6+x^7+x^8+x^9$
$+x^{10}+x^{13}+x^{15}+x^{16}$</td><td>$\underline{(1+x)}.\ (1+x+x^3+x^5+x^7).$
$(1+x+x^2+x^4+x^5+x^6+x^8)$</td><td colspan="3"></td></tr>
<tr><td colspan="3"></td><td>$G_3(x)$</td><td>$1+x+x^2+x^3+x^5+x^6+x^9$
$+x^{10}+x^{12}+x^{14}+x^{15}+x^{16}$</td><td>$\underline{(1+x)}\ .\ \underline{(1+x+x^7)}\ .$
$(1+x+x^5+x^6+x^8)$</td></tr>
<tr><td>$G_4(x)$</td><td>$1+x+x^6+x^{10}$
$+x^{12}+x^{16}$</td><td>$\underline{(1+x)}.(1+x+x^2+x^3+x^7)\ .$
$(1+x+x^4+x^5+x^6+x^7+x^8)$</td><td colspan="3" rowspan="2"></td></tr>
<tr><td>$G_5(x)$</td><td>$1+x^5+x^6+x^7+x^8+x^9$
$+x^{10}+x^{16}$</td><td>$\underline{(1+x)}.(1+x^3+x^7).$
$(1+x+x^2+x^5+x^6+x^7+x^8)$</td></tr>
<tr><td colspan="3"></td><td>$G_6(x)$</td><td>$1+x^3+x^6+x^7+x^{10}$
$+x^{13}+x^{14}+x^{16}$</td><td>$\underline{(1+x)}.\underline{(1+x+x^7)}\ .$
$(1+x^2+x^3+x^4+x^5+x^7+x^8)$</td></tr>
</table>

Tableau 4.2 - Les polynômes générateurs utilisés

4.4.1 Pouvoir intrinsèque de détection d'erreurs des polynômes générateurs

Cette section présente le pouvoir intrinsèque de détection des polynômes générateurs décrits dans le tableau 4.2, face à des erreurs générées de façon aléatoire ou exhaustive. Le but est de représenter la différence entre les pouvoirs de détection d'erreurs selon un bon ou un mauvais choix de polynômes générateurs.

Avec ces polynômes générateurs, nous avons lancé plusieurs séries d'expérimentions avec 10^7 messages erronés multiples de G_1, générés de façon aléatoire ou exhaustive, dans les modèles de simulation présentés aux figures 4.10 et 4.11.

À la fin de cette série d'expériences, le nombre des valeurs booléennes à "1" dans les blocs respectifs *CRC_syndrome_i* (i allant de 2 à 6) représente le nombre de messages erronés et non détectés par les polynômes générateurs correspondants G_i. Ces nombres sont les entrées des blocs respectifs *Frame-Error-Rate-Calculation*, qui fournissent les taux de non détection intrinsèques des messages erronés par chacun des polynômes générateurs G_i.

Ces taux sont représentés à la figure 4.12, en fonction du polynôme générateur utilisé.

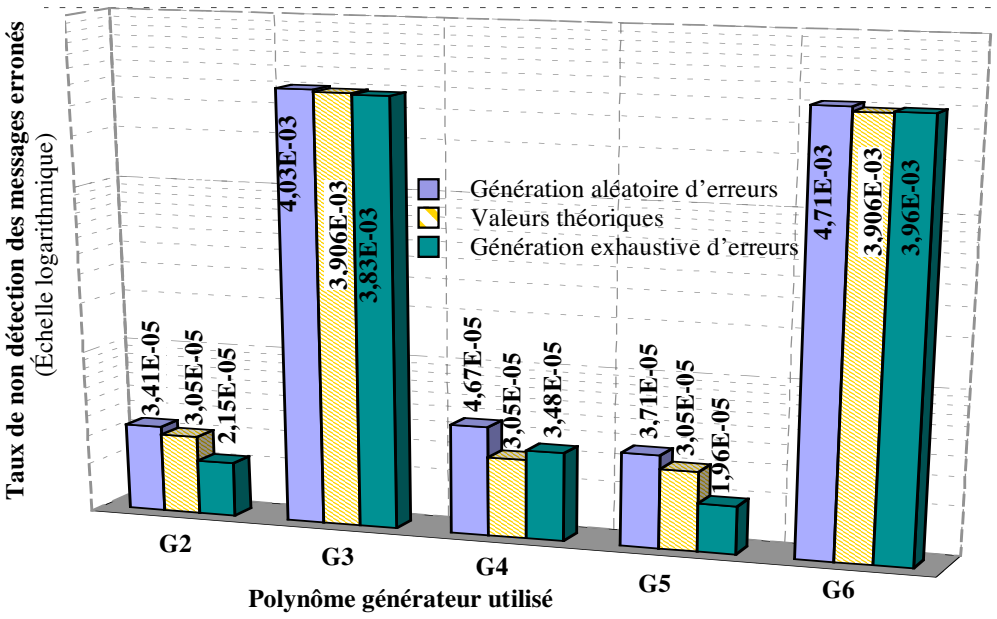


Figure 4.12 - Taux intrinsèque de non détection des messages erronés par G_i ($i \in [2, 6]$)

Ces résultats de simulation valident les résultats théoriques. En effet, pour un taux théorique de non détection des messages erronés multiples de G_1, par G_2 ou G_4 ou G_5 égal à 2^{-15}, soit $3{,}05.10^{-5}$ (ces trois polynômes diffèrent de G_1 d'un facteur polynomial de degré 15), les modèles de simulation donnent pour G_2 des taux respectifs de $3{,}14.10^{-5}$ et $2{,}15.10^{-5}$ pour les générations aléatoire et exhaustive des erreurs. Pour G_4, les taux respectifs sont de $4{,}67.10^{-5}$ et $3{,}48.10^{-5}$, tandis que pour G_5, ils sont de $3{,}71.10^{-5}$ et $1{,}96.10^{-5}$.

Concernant les polynômes générateurs G_3 ou G_6, qui diffèrent de G_1 par un facteur polynomial de degré 8 seulement, le taux théorique de non détection des messages erronés est de 2^{-8} soit $3{,}9.10^{-3}$, et les modèles de simulations donnent pour G_3 des taux respectifs de $4{,}03.10^{-3}$ et $3{,}83.10^{-3}$ pour les générations aléatoire et exhaustive. Pour G_6, les taux respectifs sont de $4{,}71.10^{-3}$ et $3{,}96.10^{-3}$.

Par la suite, l'apport de la fonction de contrôle d'erreur évolutive, en termes de pouvoir de détection d'erreurs, qui dépend du nombre de polynômes générateurs qui la composent et de leur pouvoir respectif de détection, est validé de deux façons. Dans le premier cas, nous déterminons le taux de non détection des messages erronés multiples de G_1 par au moins un des polynômes générateurs. Ce sera la section 4.4.2. Dans le second cas, nous déterminons le taux de non détection par l'ensemble de tous les polynômes générateurs. Ce sera la section 4.4.4.

4.4.2 Pouvoir de détection des erreurs par au moins un des polynômes générateurs formant la fonction de contrôle

Nous comparons dans cette section le pouvoir de détection par au moins un des polynômes générateurs qui composent la fonction de contrôle évolutive, par rapport au pouvoir de détection dans le cas idéal, supposé traduire le niveau maximal de détection des erreurs.

Ce cas idéal est tel que tous les messages erronés non détectables par G_1, sont détectés par **tous** les autres polynômes générateurs utilisés. Par exemple, le pouvoir de détection par le changement cyclique entre deux polynômes générateurs (ex. : G_1 et G_2) serait de 50% dans le cas idéal, vu qu'à la base, la moitié des messages erronés injectés dans les modèles de simulation sont non détectés par G_1. L'apport serait de 66,6 % avec trois polynômes générateurs (ex. : G_1, G_2 et G_3), de 75% avec quatre polynômes générateurs, et ainsi de suite.

La figure 4.13 présente les différences, par rapport au cas idéal, de plusieurs changements cycliques entre un nombre variable de polynômes générateurs, et cela pour les générations aléatoire et exhaustive des messages erronés multiples de G_1.

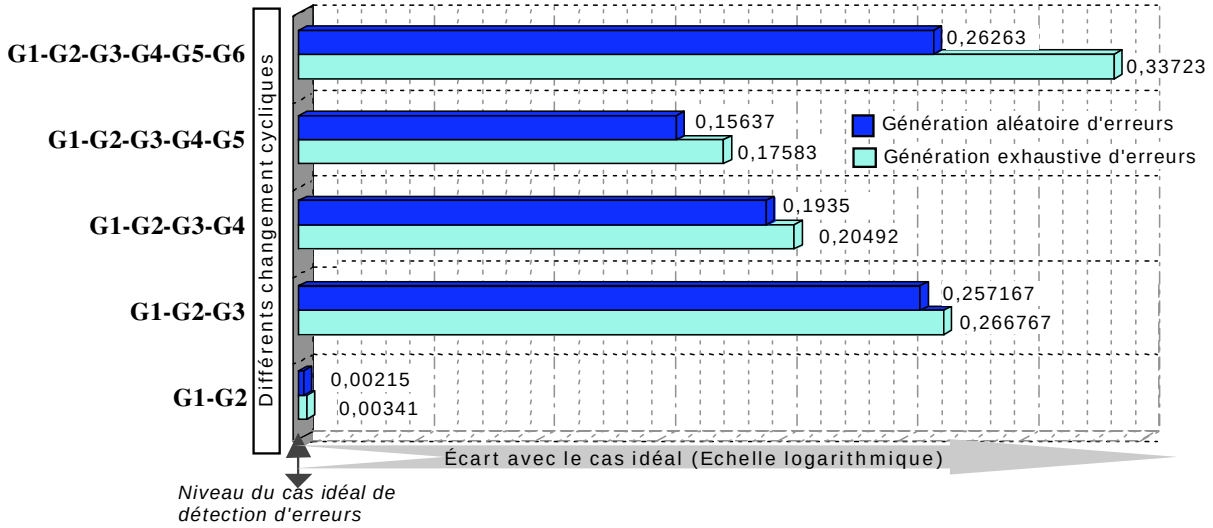


Figure 4.13 - Écarts entre le pouvoir de détection idéal et celui observé par simulation (avec générations aléatoire et exhaustive de messages erronés)

Le premier constat est qu'avec deux polynômes générateurs (G_1 et G_2), les simulations donnent des résultats très proches du cas idéal ; les écarts étant respectivement de 0,00215 et 0,00341 pour les générations exhaustive et aléatoire. Ensuite, avec trois polynômes générateurs (G_1, G_2 et G_3), le pouvoir de détection d'erreurs diminue considérablement, et ce pour les deux types de génération des messages erronés. En effet, l'écart s'est accru d'un facteur de l'ordre de la centaine. Ensuite, le pouvoir de détection augmente à nouveau (de façon monotone) en ajoutant successivement les polynômes G_4 puis G_5 et enfin il re-diminue en ajoutant le polynôme G_6.

Ces variations des pouvoirs de détection d'erreurs selon chacune des combinaisons de polynômes générateurs utilisés sont cohérentes avec la théorie. L'utilisation de G_3 provoque une diminution du pouvoir de détection, car G_3 a un facteur commun avec G_1 beaucoup plus important que G_2, contrairement à G_4 et G_5 qui permettent d'augmenter à nouveau le pouvoir de détection, mais sans arriver à compenser la perte occasionnée par G_3. L'effet de G_6 s'explique comme celui de G_3.

4.4.3 Génération exhaustive ou génération aléatoire des erreurs ?

Dans cette section, nous allons interpréter les résultats présentés sur les figures 4.12 et 4.13, selon le type de génération aléatoire ou exhaustive des messages erronés, dans le but de choisir le type de génération qui correspond le mieux à nos besoins en termes de validation.

Des résultats reportés sur la figure 4.12, se déduisent trois constats :

1) le taux de non détection des messages erronés est plus important avec une génération aléatoire qu'avec une génération exhaustive,
2) le pouvoir de détection des messages erronés est plus proche du pouvoir de détection théorique avec une génération exhaustive qu'avec une génération aléatoire,
3) le pouvoir de détection des messages erronés avec une génération exhaustive est même parfois supérier au pouvoir théorique, ce qui est incohérent à première vue.

Le premier constat peut avoir, selon nous, deux explications. La première est qu'avec une génération aléatoire des messages erronés, il n'est pas exclu que plusieurs de ces messages soient identiques et non détectés par le polynôme générateur utilisé. La deuxième est, qu'avec la génération exhaustive des messages erronés, le mécanisme utilisé ne permet en fait de générer qu'une partie de ces messages (erreurs ne portant que sur les 32 bits de poids faible). De plus, cette génération est limitée à 10^7 messages erronés, donc seuls 23 des 32 bits de poids faible de ces messages sont réellement concernés. La génération exhaustive, telle que nous l'avons modélisée, n'est donc pas une génération totalement exhaustive.

La génération exhaustive des erreurs est donc particulière, vu qu'elle ne couvre qu'une partie des messages erronés possibles. En fait, cette particularité de la génération exhaustive d'erreurs explique aussi l'incohérence de certains résultats de la figure 4.12, et le troisième constat relevé ci-dessus, où le pouvoir de détection d'erreurs est, pour certains polynômes générateurs, supérieur au pouvoir théorique. C'est par exemple le cas de G_2 et G_5, où pour un taux théorique de non détection d'erreurs de $3{,}05.10^{-5}$, les modèles de simulation donnent un taux de non détection d'erreurs de $2{,}15.10^{-5}$ pour G_2, et de $1{,}96.10^{-5}$ pour G_5.

Des résultats reportés sur la figure 4.13, nous remarquons que, pour chacune des combinaisons de polynômes générateurs utilisée, la génération exhaustive des erreurs garantit un pouvoir de détection plus proche du cas idéal que pour une génération aléatoire.

Nous pouvons tirer deux conclusions des résultats de la génération exhaustive des erreurs :

1) ces résultats ne sont pas toujours assez crédibles, notamment car pouvant apparaître meilleurs que ceux théoriques,
2) ces résultats sont toujours meilleurs que ceux observés avec la génération aléatoire ; c'est donc cette dernière qui est donc à privilégier lorsqu'on cherche à connaître les pire cas, donc à être pessimiste.

Ainsi, la génération aléatoire d'erreurs est à la fois plus réaliste et plus sécuritaire. En effet, elle est mieux représentative des types d'erreurs de transmission considérés, en particulier de ceux relatifs au mode de défaillance du nœud intermédiaire provoquant la transmission répétitive de messages erronés non détectés par la fonction de contrôle d'erreur utilisée. De plus, la représentativité de ce mode de défaillance du nœud intermédiaire par les modèles de simulations se base sur des inversions de bits sur toute la longueur du message généré.

Par ailleurs, l'utilisation d'une génération aléatoire présente un autre avantage important : celui d'un temps d'exécution des simulations plus court (de l'ordre de trois fois plus faible que pour une génération exhaustive). Le temps d'exécution des modèles de simulation générant de façon aléatoire 10^7 messages erronés multiples de G_1 est de 17 heures et 43 minutes, contre 2 jours 3 heures et 14 minutes avec une génération exhaustive.

En conclusion, pour toutes ces raisons, nous privilégions par la suite les modèles de simulation basés sur une génération aléatoire des erreurs.

4.4.4 Pouvoir de détection d'une erreur par tous les polynômes générateurs formant la fonction de contrôle

Nous allons modifier ces modèles dans le but de valider d'une autre manière l'apport en termes de détection d'erreurs du changement cyclique entre les polynômes générateurs. En effet, nous cherchons, par la suite, à calculer le taux de non détection d'un **même** message erroné par **tous** les polynômes générateurs, alors que jusqu'à présent nous avons cherché à calculer le taux de non détection des messages erronés par **au moins un** des polynômes générateurs.

Et dans le cadre de cette nouvelle analyse, nous comparons également les pouvoirs de détection d'erreurs dans le cas de changements cycliques entre des polynômes générateurs standard et non standard [Witzke & Leung 1985].

Nous commençons par décrire les changements apportés aux modèles de simulation basés sur une génération aléatoire des messages erronés. Puis, nous montrons l'apport de la fonction de contrôle évolutive en termes de pouvoir de détection d'erreurs.

4.4.4.1 Description des changements apportés aux modèles de simulation

Les changements apportés aux modèles de simulation sont décrits ci-après et représentés sur la figure 4.14. Ils concernent la détection d'erreurs, et sont donc réalisés à la réception.

1er changement : sur l'injection d'erreurs. La comparaison entre les pouvoirs de détection des m polynômes générateurs utilisés "*Gi*" ($i \in [1, n]$) s'effectue désormais en injectant les **mêmes** messages erronés, multiples de G_1, à tous les blocs *General_CRC Syndrome Detector-i* relatifs aux polynômes G_i. Par rapport au précédent modèle de simulation, la période de génération des messages erronés est ainsi multipliée par le nombre de polynômes générateurs utilisés.

2ème changement : sur le processus de détection d'un message erroné. Dans les précédents modèles, la sortie d'un bloc *General_CRC Syndrome Detector-i* était directement reliée à l'entrée d'un bloc *CRC_Syndrome_i*. Ainsi, le nombre de valeurs booléennes à 0 collectées par ce bloc correspondait au nombre de messages erronés et non détectés par le polynôme G_i.

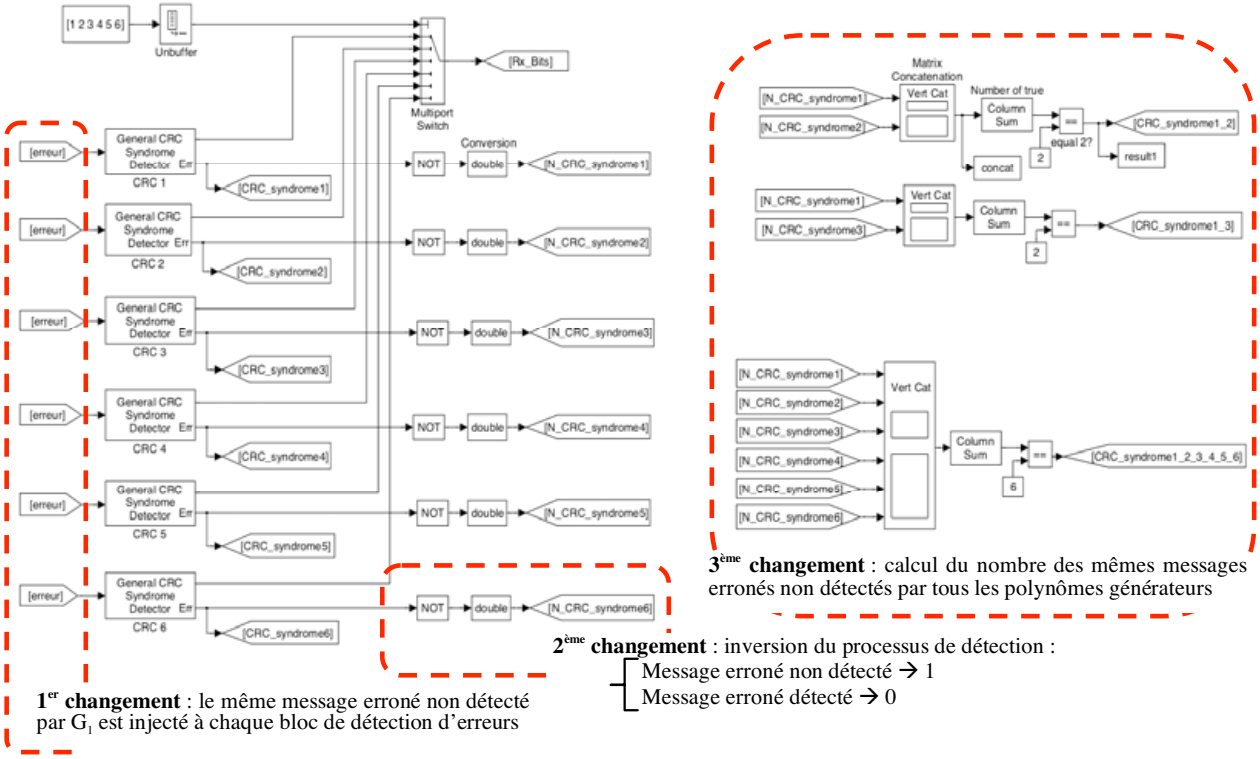
Dans les nouveaux modèles, nous intercalons entre ces deux blocs, un bloc d'inversion des valeurs booléennes, afin que l'occurrence de la non détection d'un message corresponde à l'écriture de la valeur "1" dans le bloc *CRC_Syndrome_i*, renommé *N-CRC Syndrome*-i.

3ème changement : sur la détermination du nombre de messages erronés. Ce changement est lié au précédent et cible la détermination du nombre de messages erronés et non détectés par G_1 ainsi que par un ou plusieurs autres polynômes générateurs G_i. Par exemple, pour calculer le nombre de messages erronés et non détectés à la fois par G_1 et G_2 et … et G_m, nous échantillonnons les valeurs booléennes collectées dans chacun des blocs *N-CRC Syndrome-i (i* allant 1 à *m)*, puis additionnons ces valeurs échantillonnées.

Ainsi, une valeur échantillonnée égale à 1 correspond à la non détection d'un message erroné alors que dans les modèles de simulations précédents, une valeur égale à 1 correspondait à la détection d'un message erroné. Une somme des valeurs échantillonnées égale au nombre m de polynômes générateurs G_i (i allant de 1 à m), correspond donc à un message erroné non détecté par tous ces polynômes et se traduit par l'écriture d'une valeur booléenne égale à "1" dans le bloc *CRC_syndrome 1_2_…_m*.

À la fin des simulations, et après injection de 10^7 messages erronés multiples de G_1, la somme des valeurs booléennes à "1" dans le bloc *CRC_syndrome 1_2_…_m* correspond au nombre de messages erronés et non détectés à la fois par G_1, G_2 et … et G_m.

Dans la suite, nous ne présentons que des simulations basées essentiellement sur des combinaisons de deux polynômes générateurs (dont toujours G_1), et donc sur des blocs *CRC_Syndrome_1_i*.



1er changement : le même message erroné non détecté par G_1 est injecté à chaque bloc de détection d'erreurs

2ème changement : inversion du processus de détection :
- Message erroné non détecté → 1
- Message erroné détecté → 0

3ème changement : calcul du nombre des mêmes messages erronés non détectés par tous les polynômes générateurs

Figure 4.14 - Nouveaux changements apportés aux modèles de simulations pour évaluer le pouvoir de détection par tous les polynômes

4.4.4.2 Apport du changement cyclique entre des polynômes générateurs

Les nouveaux modèles de simulation nous permettent, en plus de confirmer, en termes de pouvoir de détection d'erreurs, l'apport du changement cyclique entre les polynômes générateurs, d'affiner sa portée en introduisant une comparaison entre une utilisation de polynômes générateurs standard et non standard.

En effet, dans ces nouveaux modèles, nous utilisons 6 polynômes générateurs standard, et deux polynômes générateurs non standard déjà utilisés dans les modèles précédents. Ces polynômes, décrits dans le tableau 4.3, ont des pouvoirs de détection complémentaires avec le polynôme générateur G_1.

<table>
<tr><td colspan="5">$G_1(x) = \underline{(1+x)}.(1+x+x^7).(1+x^2+x^3+x^4+x^8)$</td></tr>
<tr><td colspan="3">Polynômes générateurs standard</td><td colspan="2">Polynômes générateurs non standard</td></tr>
<tr><td colspan="3">$G_i(x)$ = (1+x)*un polynôme primitif de degré 15 (sauf G2(x))</td><td colspan="2">$G_i(x)$ = (1+x)* un polynôme primitif de degré 7 * un polynôme primitif de degré 8</td></tr>
<tr><td rowspan="2">Identifiant</td><td colspan="2">Forme</td><td rowspan="2">Identifiant</td><td colspan="2">Forme</td></tr>
<tr><td>développée</td><td>décomposée</td><td>développée</td><td>décomposée</td></tr>
<tr><td>$G_2(x)$ = IEEE-WG77.1</td><td>$1+x+x^5+x^6+x^8+x^9+x^{10}+x^{11}+x^{13}+x^{14}+x^{16}$</td><td>$(1+x^2+x^3+x^4+x^8).(1+x+x^2+x^4+x^5+x^6+x^8)$</td><td colspan="3"></td></tr>
<tr><td>$G_3(x)$ = CRC-CCITT</td><td>$1+x^5+x^{12}+x^{16}$</td><td>$\underline{(1+x)}.(1+x+x^2+x^3+x^4+x^{12}+x^{13}+x^{14}+x^{15})$</td><td colspan="3"></td></tr>
<tr><td colspan="3"></td><td>$G_4(x)$</td><td>$1+x+x^6+x^{10}+x^{12}+x^{16}$</td><td>$\underline{(1+x)}.(1+x+x^2+x^3+x^7).(1+x+x^4+x^5+x^6+x^7+x^8)$</td></tr>
<tr><td>$G_5(x)$ = IBM-SDLC</td><td>$1+x+x^2+x^4+x^7+x^{13}+x^{15}+x^{16}$</td><td>$\underline{(1+x)}.(1+x^2+x^3+x^7+x^8+x^9+x^{10}+x^{11}+x^{12}+x^{15})$</td><td colspan="3"></td></tr>
<tr><td colspan="3"></td><td>$G_6(x)$</td><td>$1+x^5+x^6+x^7+x^8+x^9+x^{10}+x^{16}$</td><td>$\underline{(1+x)}.(1+x^3+x^7).(1+x+x^2+x^5+x^6+x^7+x^8)$</td></tr>
<tr><td>$G_7(x)$ = CRC-16Q*</td><td>$1+x+x^3+x^4+x^5+x^6+x^8+x^{11}+x^{15}+x^{16}$</td><td>$\underline{(1+x)}.(1+x+x^3+x^5+x^8+x^9+x^{10}+x^{15})$</td><td colspan="3"></td></tr>
<tr><td>$G_8(x)$ = CRC-16</td><td>$1+x^2+x^{15}+x^{16}$</td><td>$\underline{(1+x)}.(1+x+x^{15})$</td><td colspan="3"></td></tr>
<tr><td>$G_9(x)$ = IEC-TC57</td><td>$1+x+x^4+x^7+x^8+x^9+x^{11}+x^{12}+x^{14}+x^{16}$</td><td>$\underline{(1+x)}.(1+x^4+x^5+x^6+x^8+x^{11}+x^{14}+x^{15})$</td><td colspan="3"></td></tr>
</table>

Tableau 4.3 - Descriptif des polynômes générateurs utilisés

Suite à l'injection dans le modèle de simulation de 10^7 messages erronés multiples de G_1, nous récupérons, à partir des blocs *CRC_Syndrome_1_i* (i = 2 à 9), les valeurs à "1" dont la somme correspondant au nombre de messages erronés et non détectés par tous les polynômes générateurs G_i (*i* allant de 1 à 8).

Nous représentons ensuite les évolutions du nombre et du taux des messages erronés, et non détectés par le changement cyclique entre les polynômes générateurs *"G_1-G_i"*, suivant le nombre des messages erronés injectés (cf. figure 4.15).

Pour des raisons de lisibilité du comparatif entre les pouvoirs de détection d'erreurs apportés par les polynômes générateurs standard et non standard, nous avons choisi de ne représenter les résultats que pour trois des six polynômes générateurs standard en plus des deux polynômes générateurs non standard G_4 et G_6. Il s'agit du polynôme générateur standard G_2 qui est un polynôme primitif de degré 16 et des polynômes générateurs standard G_3 et G_5 qui représentent le mieux les évolutions des pouvoirs de détection d'erreurs : respectivement le plus grand et le plus petit nombre de messages erronés non détectés.

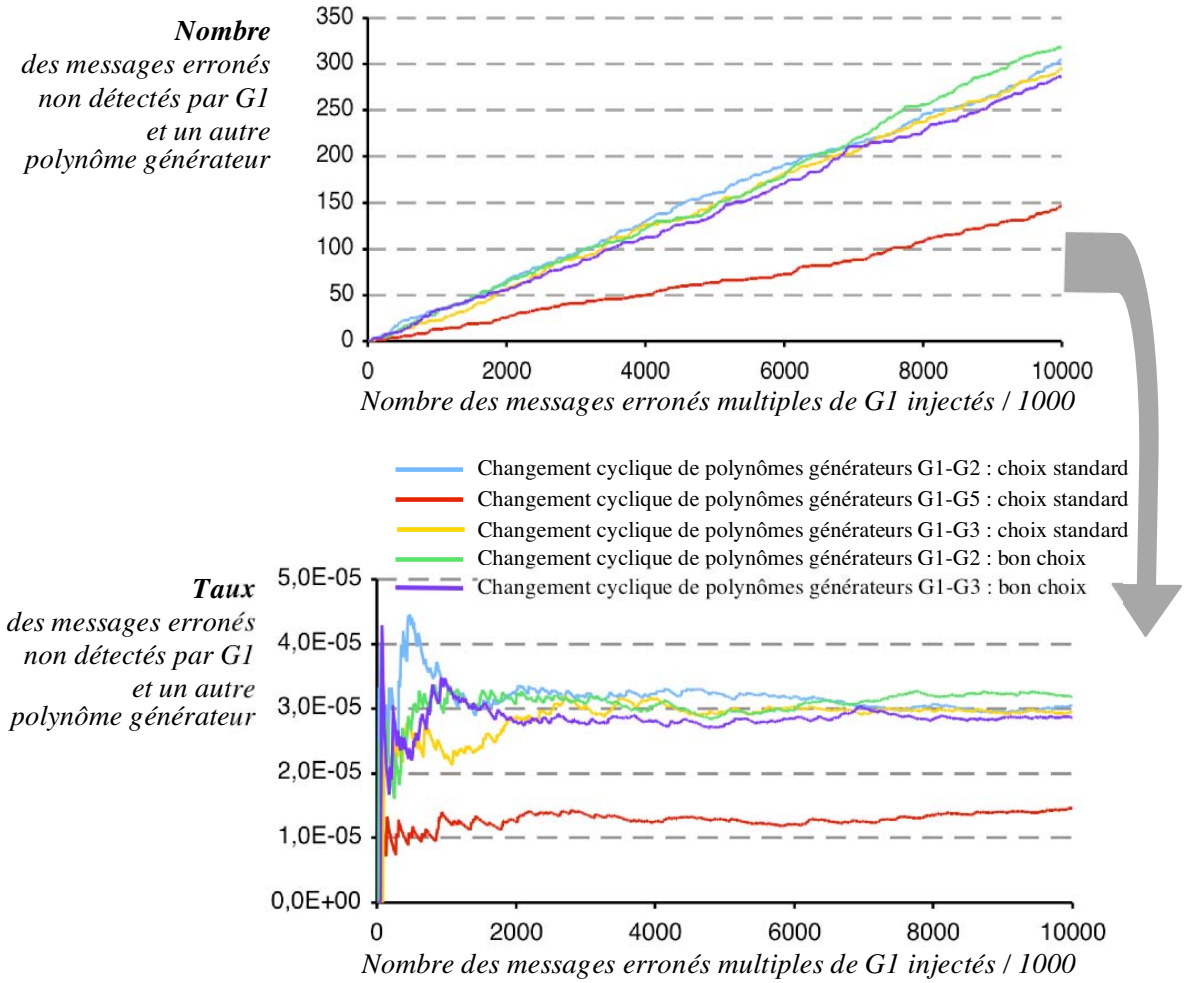


Figure 4.15 - Comparaison entre les pouvoirs de détection d'erreurs par les changements cycliques entre des polynômes générateurs standard et non standard

La première remarque est que les pouvoirs de détection d'erreurs apportés par les changements cycliques entre des polynômes générateurs standard et non standard sont équivalents. En effet, sur les 10^7 messages erronés multiples de G_1 injectés, les nombres de messages erronés et non détectés par G_1 et par le polynôme générateur non standard G_4, ou par G_1 et par le polynôme générateur non standard G_6, sont respectivement de 305 et 296, tandis que les nombres de messages erronés et non détectés par G_1 et par le polynôme générateur standard G_5, ou par G_1 et par le polynôme générateur standard G_3, sont respectivement de 286 et 318.

L'utilisation de polynômes générateurs standard ou non n'influe donc pas de manière déterminante sur le pouvoir de détection d'erreurs de la fonction de contrôle évolutive. Le seul critère important pour le choix de ces polynômes est la complémentarité entre leurs pouvoirs de détection d'erreurs.

La deuxième remarque est que le pouvoir de détection d'erreurs du changement cyclique entre G_1 et le polynôme générateur standard G_2 est pratiquement deux fois plus important que celui des autres changements cycliques. En effet, le nombre de messages erronés non détectés par G_1 et G_2 est de 146, contre une moyenne de 300 messages avec les autres changements cycliques. Ceci est dû à la particularité du polynôme générateur standard G_2 qui ne présente aucun polynôme primitif en commun avec G_1 tandis que les autres polynômes générateurs ont le polynôme primitif (1+x) en commun avec G_1.

La troisième remarque est que les nouveaux modèles de simulation valident, eux aussi, l'apport théorique en termes de pouvoir de détection d'erreurs de la fonction de contrôle évolutive. En effet, les résultats de la simulation donnent un taux de non détection d'erreurs par le changement cyclique entre G_1 et le polynôme générateur standard G_2 très proche de la valeur théorique de $2^{-16} = 1{,}5.10^{-5}$, et des taux proches de la valeur théorique de $2^{-15} = 3.10^{-5}$ pour les autres changements.

Maintenant, en se basant sur ces résultats de simulation, la dernière partie de ce chapitre a pour but de valider l'apport de la fonction de contrôle d'erreur évolutive par rapport à l'objectif d'intégrité de haut niveau pour les systèmes de CDV : taux d'embarquement des surfaces de contrôle inférieur à 10^{-9}/h, et pouvant résulter de la non détection de X messages erronés dans un lot de N messages ($X = 3$ et $N = 10$).

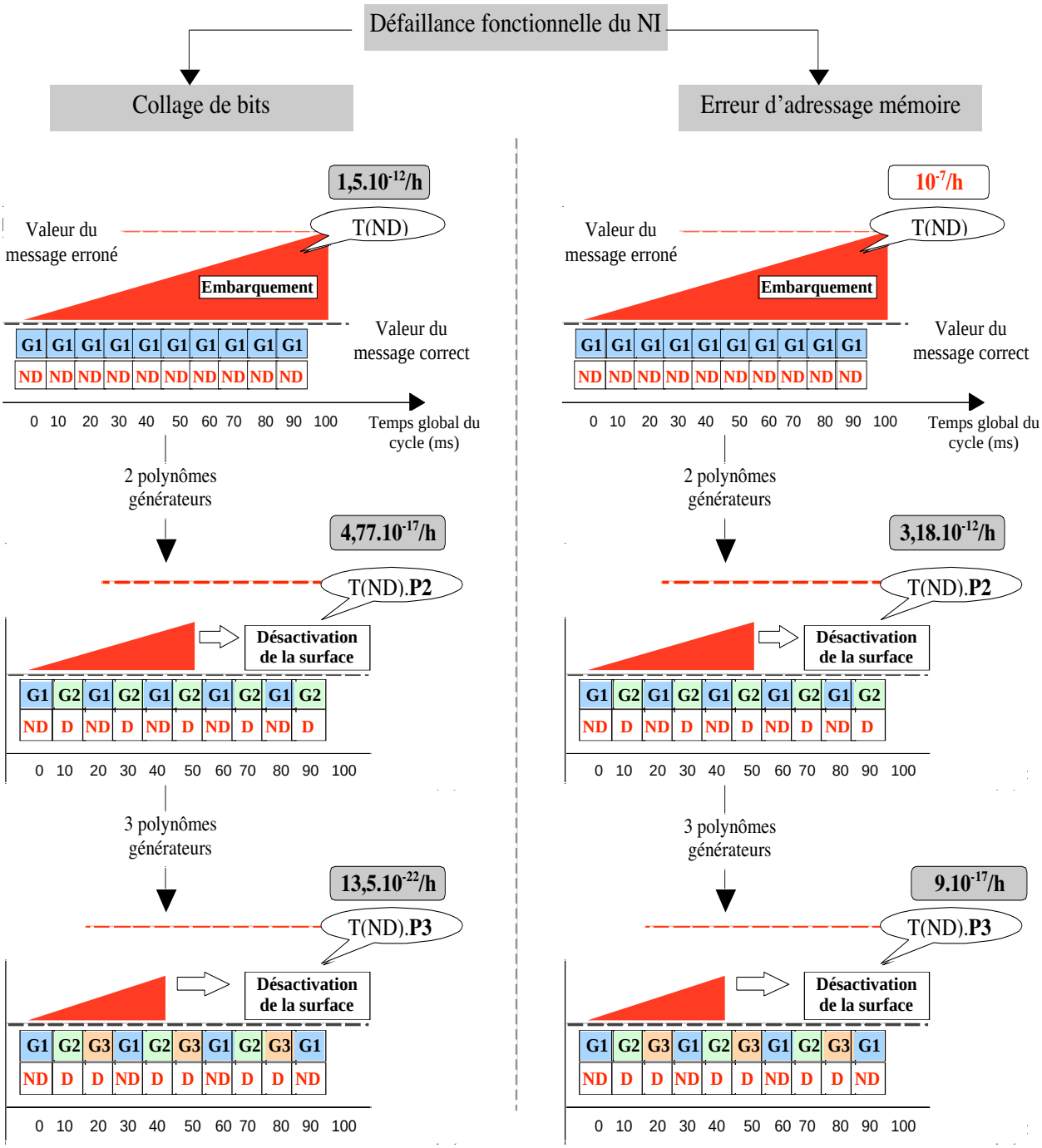
4.5 Validation de la fonction de contrôle dans le cas des systèmes de CDV

Nous rappelons que dans le cas des systèmes de CDV, lorsqu'une surface de contrôle traite des commandes erronées et non détectées pendant 100 ms, cela peut conduire à l'embarquement de cette surface. Avec un rafraîchissement périodique de la surface de contrôle tous les 10 ms, la durée au bout de laquelle son embarquement pourrait se produire correspond donc à la réception de 10 commandes qui seraient erronées et non détectées. L'étude des risques réalisée au chapitre 2, sur la base de codes CRC 16 bits, montre que les modes de défaillances pouvant conduire à un taux d'embarquement jugé non satisfaisant sont liés à certaines fonctionnalités des nœuds intermédiaires.

En effet, pour le premier mode de défaillance étudié, qui représente le *collage de bits* au niveau du tampon de stockage de données du nœud intermédiaire, le taux d'embarquement des surfaces de contrôle est égal à $1{,}5.10^{-12}$/h. Pour le deuxième mode de défaillance, qui concerne les *erreurs d'adressage mémoire,* le taux d'embarquement est égal au taux de défaillance fonctionnelle du nœud intermédiaire fixé dans le cadre de notre cas d'étude à 10^{-7}/h.

Une des conclusions du chapitre 3 était, qu'en appliquant le changement cyclique entre plusieurs fonctions de contrôles, et en choisissant une stratégie de recouvrement adéquate, nous pouvions considérablement diminuer le taux d'embarquement des surfaces de contrôle. Toutefois, nous n'avions pas encore déterminé dans ce cas les nouvelles valeurs des taux d'embarquement nous permettant de les comparer au seuil de criticité de 10^{-9}/h.

Dans la figure 4.16, nous présentons les valeurs des taux d'embarquement de la surface de contrôle après l'application du changement cyclique entre deux puis trois polynômes générateurs, et ceci pour les deux modes de défaillances considérés pour le nœud intermédiaire.



T(ND) = Taux de la défaillance fonctionnelle du NI entraînant la transmission de messages erronés et non détectés par G1
P2 = Probabilité de non-détection du message erroné par G1 et G2
P3 = Probabilité de non-détection du message erroné par G1 et G2 et G3

Figure 4.16 - Apport du changement cyclique de fonctions de contrôle dans le cas des systèmes de CDV

Nous remarquons qu'en présence des erreurs d'adressage de la mémoire, le taux d'embarquement de la surface de contrôle passe de 10^{-7}/h à $3{,}18.10^{-12}$/h après application du changement cyclique entre G_1 et G_2, satisfaisant ainsi l'objectif d'intégrité de haut niveau visé. Ce taux diminue encore considérablement en utilisant trois polynômes générateurs : il passe à 9.10^{-17}/h.

En présence de défaillances "*collage de bits*", l'objectif d'intégrité était déjà satisfait avec un seul polynôme générateur. En utilisant le changement cyclique entre G_1 et G_2, le taux d'embarquement de la surface de contrôle diminue et passe de $1{,}5.10^{-12}$/h à $4{,}77.10^{-17}$/h ; en utilisant G_1, G_2 et G_3, ce taux passe à $1{,}35.10^{-21}$/h.

Nous montrons ainsi, en se basant sur les résultats des simulations, que l'application de la fonction de contrôle évolutive garantit des taux d'embarquement des surfaces de contrôle satisfaisant aux objectifs d'intégrités fixés.

4.6 Conclusion

Ce chapitre a d'abord décrit une mise en œuvre de la fonction de contrôle évolutive proposée au chapitre 3, ceci en se basant sur des codes CRC. Sur ce point, nous avons montré qu'une mise en œuvre logicielle au niveau applicatif est tout à fait compatible avec les contraintes temps réel imposées par son utilisation dans les systèmes de CDV. Nous avons aussi décrit les bases mathématiques sur lesquelles nous nous appuyons pour définir les différents polynômes générateurs qui constituent la fonction de contrôle évolutive : il s'agit de la décomposition d'un polynôme générateur en un produit de polynômes primitifs.

La suite du chapitre a été consacrée à la validation par simulation de la mise en œuvre présentée dans ce chapitre. Sur la base de l'outil *Matlab-Simulink*, nous avons présenté des conceptions et des simulations de modèles qui nous ont permis d'évaluer le pouvoir de détection d'erreurs de transmission assuré par un changement cyclique entre des polynômes générateurs. Nous avons également présenté les évolutions de ce pouvoir de détection d'erreurs suivant le nombre de polynômes générateurs utilisés, la façon de les alterner, et la complémentarité de leurs pouvoirs de détection d'erreurs.

Nous avons également cherché à réduire la durée des simulations en optimisant la manière d'introduire des erreurs de transmission dans les modèles. Pour ce faire, d'une part, nous n'avons considéré que des erreurs qui sont multiples d'un des polynômes générateurs et donc non détectées par celui-ci, et d'autre part, nous avons utilisé une génération aléatoire des erreurs qui est plus réaliste qu'une démarche exhaustive et moins coûteuse en temps d'exécution.

Les résultats que nous avons obtenus au cours des différentes simulations ont confirmé l'apport escompté de la fonction de contrôle évolutive en termes de pouvoir de détection de messages erronés au sein d'un lot de messages. Au cours de différentes simulations, nous avons aussi montré l'influence d'un mauvais choix des polynômes générateurs, c'est-à-dire l'utilisation de polynômes générateurs ayant des polynômes primitifs en commun.

Finalement, nous avons montré, en se basant sur les résultats des simulations, que l'application aux systèmes de CDV, du changement cyclique entre les polynômes générateurs, conduit à un taux d'embarquement des surfaces de contrôle qui, pour tous les types des risques considérés, est nettement inférieur au seuil de criticité de 10^{-9}/h.

Chapitre 5 - Microsystèmes : état de l'art et perspectives d'intégration dans des SCC

L'utilisation massive de microsystèmes dans les SCC a été le fil directeur sous-jacent de nos travaux sur un nouveau système de communication adéquat, plus particulièrement en termes d'intégrité pour les messages échangés. Dans le premier chapitre, nous avions donné un premier aperçu des différentes définitions et problématiques des microsystèmes, et justifié pourquoi, dans le contexte de nos travaux, nous ne retenions principalement que le critère du *nombre* de microsystèmes. Les bases d'un tel nouveau système de communication ayant été établies, il est maintenant possible, dans ce dernier chapitre, de se pencher plus précisément sur l'impact de l'intégration de microsystèmes dans les SCC, en vue de commander des centaines, des milliers, ou plus encore, de microactionneurs.

Nous commençons par dresser un état de l'art plus détaillé des caractéristiques des microsystèmes individuels, en nappes et en réseaux. Il ne s'agit en aucun cas d'être exhaustif, mais de mettre en lumière, pour situer nos travaux, l'extrême diversité des problématiques et des défis couverts aujourd'hui par le concept de microsystèmes. En effet, si la base de ce concept reste principalement l'association de microcomposants électroniques et mécaniques, ces derniers sont bien souvent la traduction de la mise en œuvre de principes d'autres domaines : optique, thermique, etc. Donc, bien d'autres domaines et spécialistes sont impliqués dans la conception et l'utilisation des microsystèmes. Nous présentons des exemples, notamment dans le domaine aéronautique, ainsi que les grandes lignes des avantages et inconvénients des microsystèmes.

Nous développons ensuite plus précisément les aspects architecturaux du réseau de communication, pour ensuite présenter des pistes pour la répartition des traitements applicatifs s'appuyant sur cette architecture. Nous présentons également des pistes sur la sûreté de fonctionnement, dont notamment les besoins ou les possibilités de moduler le niveau de cette sûreté de fonctionnement. Enfin, nous donnons des premiers éléments de validation de nos choix par rapport aux contraintes temps réel, ceci sur la base d'utilisation de protocoles synchrones. Bien entendu, les CVF, notre cas d'application, servent à illustrer nos propos.

Et, nous terminons, dans une dernière partie, par la description de ce que nous considérons comme étant des problèmes encore ouverts à la suite des travaux présentés dans ce mémoire.

5.1 MEMS et Microsystèmes individuels

5.1.1 Du microcomposant (MEMS élémentaires) au microsystème

5.1.1.1 Éléments historiques

Les microsystèmes sont nés de la constante préoccupation de vouloir réduire les dimensions des systèmes. Leur développement est devenu possible, lorsque, face aux défis technologiques de fabrication posés par la miniaturisation, a été franchie l'étape permettant de passer à l'échelle micrométrique, et donc d'implanter des **microcomposants élémentaires**.

C'est ainsi, que dans le domaine de l'électronique, avec la réalisation en 1960 d'un transistor bipolaire sur silicium, puis du transistor à effet de champ de Westinghouse en 1969 [Nathanson *et al.* 1967], s'est développée la *microélectronique*, avec des technologies de fabrication à base de microgravure (oxydation, photolithographie, masquage, passivation). Ces technologies ont permis d'exploiter les *propriétés électriques* de certains matériaux (le silicium à l'origine), pour mettre en œuvre des réalisations matérielles micrométriques du traitement d'information sous forme électrique. Cela a conduit aux circuits intégrés numériques (le cœur de la micro-informatique), puis à l'intégration de puissance, de radios fréquences et à l'optoélectronique.

Parallèlement, l'idée d'exploiter également les *propriétés mécaniques* du silicium [Thielicke & Obermeier 2000] a conduit au concept de MEMS, *Micro Electro-Mechanical System*, qui sous-entend l'intégration sur une même puce de *fonctionnalités hétérogènes*. Le développement, en particulier, des techniques de micro-usinage, et des techniques issues de la micro électromécanique, a rendu possible de réaliser sur puce des fonctionnalités de transduction ou d'actionnement (ces techniques ont notamment abouti à la réalisation de micro vannes, pompes ou moteurs). Il était dès lors possible d'associer, dans une même puce, des microcomposants mécaniques (ex. : capteurs et/ou actionneurs) et des microcomposants électroniques de traitement d'information de mesure et/ou de commande [Esteve & Campo 2004]. Par la suite, d'autres technologies de microfabrication ont été développées (moulage, soudure, collage, etc.) et d'autres matériaux que le silicium ont été employés.

Ainsi, au début des années 1970 ont été produits des capteurs de pression par gravure de plaquettes de substrat. Au début des années 1980, des expériences de micro-usinage furent menées pour créer des actionneurs utilisés dans des têtes de lecture de disque. À la fin des années 1980, le potentiel des microsystèmes devint suffisamment reconnu pour commencer à pénétrer plus avant le monde de la microélectronique et du biomédical. Dans les années 1990, la communauté européenne a créé un réseau d'excellence des microsystèmes multifonctionnels (NEXUS) promouvant la recherche et le développement des technologies MEMS. D'abord principalement consacrées au développement de microcapteurs à électronique de traitement intégré (dont les accéléromètres pour airbag), les recherches ont ensuite porté sur des microsystèmes de plus en plus intelligents, remplissant des fonctions variées. Parallèlement, aux États-Unis, l'AFOSR (Air Force Office of Scientific Research) donna son appui pour la recherche fondamentale sur les matériaux, la DARPA (Defense Advanced Research Project Agency) créa son service de fonderie en 1993, et le NIST (National Institute of Standards and Technology) entreprit de soutenir des fonderies civiles pour des dispositifs microsystèmes et des composants CMOS. À la fin des années 1990, des entreprises comme Bosch ou Motorola ont construit des usines dédiés à la production à grande échelle de microsystèmes.

5.1.1.2 Exemples

Pour donner une idée des progrès de réduction, donnons l'exemple typique, déjà très largement commercialisé dans le domaine automobile, d'un accéléromètre pour déclenchement d'airbag. La figure 5.1 illustre le taux de miniaturisation de ce système, formé d'un microcapteur et de son électronique de traitement, désormais intégrés sur une puce de 3 mm^2 [Randell & Muller 2000]. L'accéléromètre est basé sur des micropeignes fixes et mobiles imbriqués. En cas de choc, les peignes mobiles se déplacent, provoquant une variation de capacité utilisée pour engendrer le déclenchement de l'airbag.

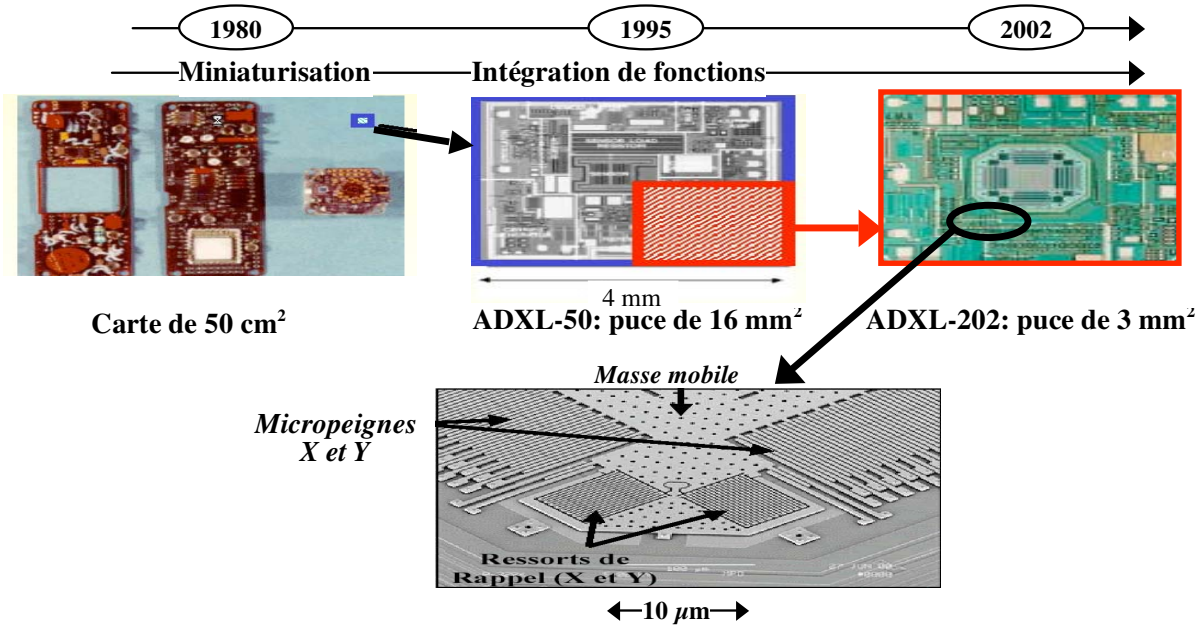


Figure 5.1 - Évolution d'un accéléromètre d'airbag [Weinberg 1999]

Toujours dans le domaine automobile, citons également l'exemple des microcapteurs de pression. Intégré dans la valve d'un pneu, un tel microcapteur permet d'alerter le conducteur d'un défaut de pression, bien avant que cela n'entraîne une difficulté de conduite.

Dans le domaine de la vidéo-projection, citons l'exemple du DMD (*Digital Mirror Device*), développé par Texas Instruments pour les vidéos projecteurs [Krishnamoorthy *et al.* 2001]. La figure 5.2 montre une vue schématique du microsystème de base, formé de deux micromiroirs avec leur mécanique d'actionnement. Chaque miroir a une surface de 16 μm^2 et gère un pixel. L'actionnement est électrostatique, avec une fréquence des mouvements pouvant atteindre 50 kHz. Plus de 1.300.000 miroirs, espacés de 1 μm, sont disposés en matrice pour constituer le dispositif complet (http://www.dlp.com/dlp/default.asp).

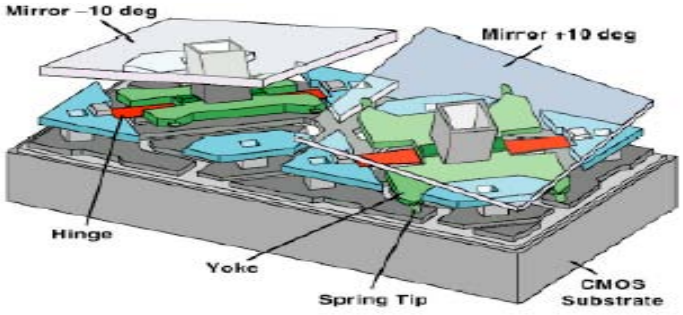


Figure 5.2 - Microsystème de base pour un système de vidéo projection

Enfin, dans le domaine qui nous intéresse plus précisément, celui de l'aéronautique, de multiples utilisations de microsystèmes sont à l'étude sans être encore au stade de la grande diffusion. Il s'agit aussi bien de microcapteurs, par exemple pour la surveillance des efforts sur la structure de l'avion, que de microactionneurs et de microsurfaces de contrôle sur les ailes d'avions, tels que des microflaps ou des microspoilers. En effet, à ce jour, les résultats de la plupart des études menées sur des drones ont montré que l'utilisation de telles microsurfaces a un effet significatif sur les couches limites et, par suite, avec des degrés variés, sur les forces de portance et de traînée appliquées sur l'avion [Fermigier 2004].

Précisons que les exemples du DMD et de ceux de l'aéronautique ne trouvent vraiment leur intérêt que dans le cadre d'une utilisation, non pas individuelle, mais en association pour former des nappes et réseaux de MEMS, développés plus loin dans ce chapitre (cf. § 5.2).

5.1.2 Définitions, diversités et classifications des microsystèmes

5.1.2.1 Définitions et vision actuelle d'un microsystème

Rappelons brièvement les variantes des définitions de MEMS et microsystèmes (cf. chapitre 1, § 1.1). Elles découlent principalement des interprétations, d'une part, du facteur de réduction d'échelle, qui est soit "*relatif*" (sur la base de technologies macroscopiques de fabrication), soit "*absolu*" (sur la base de technologies micrométriques), et d'autre part, de la constitution même d'un microsystème, qui peut aller d'un composant "*élémentaire*" (capteur, actionneur, filtre, etc.) à un système "*complet*" ou "*intelligent*", c'est-à-dire avec au moins des capacités d'actionnement et/ou de mesure et d'autonomie de traitement d'information, voire d'autonomie énergétique, et enfin des capacités de communication avec l'extérieur. Donc, aux extrêmes, nous trouvons le microcomposant absolu et le microsystème relatif.

Dans les faits, aujourd'hui, un microsystème est encore bien souvent un assemblage de composants relatifs et absolus (ex. : des microcapteurs ou microactionneurs). Mais l'objectif est bien d'intégrer à terme, de manière micrométrique, les fonctionnalités usuelles des macrosystèmes, pour obtenir des microsystèmes absolus. La figure 5.3 donne une vue fonctionnelle (classique pour un macrosystème) de ce que pourrait être un tel microsystème [Tabeling 2001].

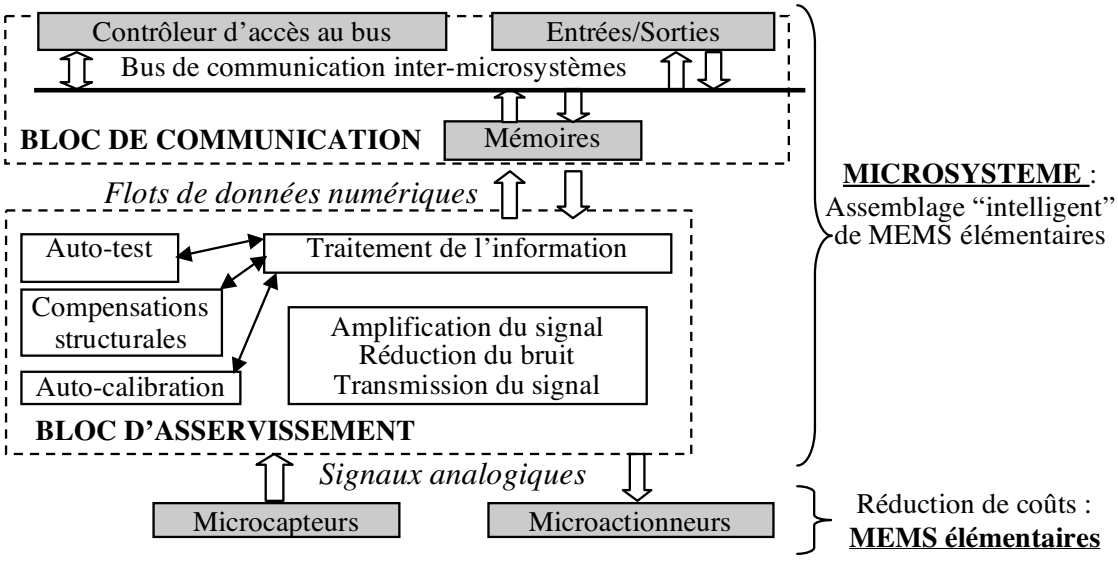


Figure 5.3 - Schéma fonctionnel d'un microsystème

Il s'agit de lire des microcapteurs et/ou de commander des microactionneurs, voire d'auto-asservir ces derniers. Le bloc "d'asservissement" comprend donc un ensemble de fonctionnalités, d'une part, de traitement des signaux de ces MEMS élémentaires (ex. : amplification), et d'autre part, de traitement des informations correspondant à ces signaux. Précisons que la figure 5.3 n'exhibe pas de fonction "énergie", pourtant inévitable dans certaines applications.

Le bloc de communication comprend les fonctionnalités permettant au microsystème d'interagir avec d'autres systèmes ou microsystèmes. Cette capacité à communiquer est très importante. Sur la figure 5.3, ces fonctionnalités se limitent à la gestion par un contrôleur, d'un bus sur lequel transitent les données du microsystème via une mémoire et des entrées/sorties. Mais ces communications, et leur mise en œuvre, peuvent être nettement plus complexes.

Enfin, indiquons deux fonctionnalités de traitement d'informations, qui sont une des priorités, voire une de spécificités, des récents développements des microsystèmes [Lyshevski 1999]. Il y a *l'autotest et l'auto-calibration* pour accroître la fiabilité et l'autonomie "d'intelligence" du microsystème, en lui permettant de se tester et calibrer lui-même. Et il y a aussi *la compensation structurale* pour améliorer les performances du microsystème, en détectant et compensant des effets d'imperfections structurales introduites lors des étapes de fabrication. En effet, le dépôt de matériau ou la gravure (ou autre) peuvent introduire des imperfections, qui se traduisent souvent par des asymétries et inélasticités des structures mécaniques, pouvant causer des perturbations permanentes (souvent des forces électrostatiques).

5.1.2.2 Diversités des microsystèmes et classifications

Aujourd'hui, si les composants de base des microsystèmes sont toujours électroniques et mécaniques, ces derniers sont néanmoins désormais bien souvent la traduction de la mise en œuvre de principes d'autres disciplines, soulevant de nombreuses problématiques et impliquant des spécialistes de divers domaines, pour la conception et pour l'utilisation des microsystèmes.

L'extrême diversité induite, dont nous ne présentons qu'un aperçu, se situe sur plusieurs plans, dont principalement : domaines d'application et secteurs économiques, disciplines scientifiques et technologies impliquées dans la conception, complexité structurelle et fonctionnelle.

Les microsystèmes trouvent désormais leur utilité dans la plupart des domaines d'application : instrumentation, périphériques (stockage de données, têtes d'impression ou de lecture de disque), contrôle de procédés, optique (microlentilles, micromiroirs), etc. D'où leur pénétration dans la plupart des secteurs économiques. Sans être exhaustif, citons :

- *l'automobile et l'aérospatial* : capteurs de navigation, de pression, de freinage, de niveau de carburant, de déclenchement de coussins gonflables de voitures,
- *les communications* : connecteurs et multiplexeurs optiques, composants radiofréquences passifs (capacités, inductances) et actifs (commutateurs, filtres), relais, en particulier dans des circuits microélectroniques pour téléphones mobiles,
- *le médical* : capteurs de pression sanguine et intracorporelle, stimulateurs musculaires et cardiaques, systèmes de diffusion intra-sanguine, prothèses, instruments d'analyses,
- *la défense* : guidage des armes, surveillance, stockage de données.

En termes de conception d'un microsystème, la pluridisciplinarité est presque toujours nécessaire. Et en considérant le spectre des domaines d'application et des secteurs économiques, presque toutes les disciplines scientifiques et les technologies associées sont impliquées : électronique, mécanique, physique, optique, thermique, fluidique, chimie, etc.

Enfin, par ailleurs, le nombre et la complexité des fonctions d'un microsystème ne cessent de croître selon l'asymptote d'évolution décrite dans la loi de Moore établie dans les années 70, qui prédit un doublement de la complexité (nombre de composants/cm^2) des circuits intégrés tous les 18 mois. La figure 5.4 décrit l'état actuel et les perspectives, en termes de nombre de transistors et de composants mécaniques dans un microsystème, et donc en termes de capacités de traitement, de capture d'information et d'actionnement [Coumar 2003].

Aujourd'hui, sont surtout bien répandus les produits avec peu de transistors et de composants mécaniques, et qui touchent des marchés "à grande diffusion", comme la téléphonie mobile ou l'automobile. À notre connaissance, le seul produit commercialisé, qui compte à la fois des milliers de transistors et de composants, est le DMD de Texas Instruments.

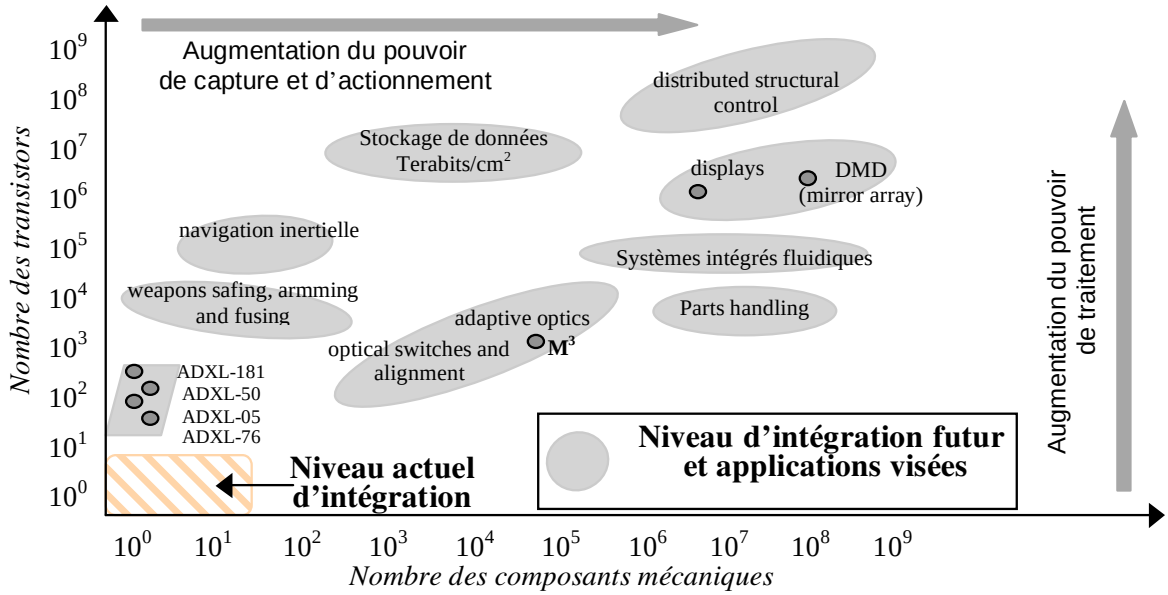


Figure 5.4 - Complexité actuelle et future des microsystèmes

De cette diversité découlent plusieurs classifications, selon les critères ou points de vue que l'on souhaite privilégier. Ainsi, il est possible d'établir des classifications par rapport aux critères suivants (certaines classifications sont issues de la prise en compte de plusieurs de ces critères) : 1) domaines d'application ou secteurs économiques, 2) technologies ou spécialités mises en jeu dans la conception d'un microsystème, 3) complexité fonctionnelle : du simple capteur ou actionneur … à plus complexe, … avec ou sans traitement d'information, 4) mais aussi d'autres critères, tels que : fixe ou mobile, passif ou actif, autonomie énergétique ou non.

5.1.3 Avantages

Si les microsystèmes suscitent un tel intérêt en termes de recherche et de marché, c'est que la miniaturisation, surtout au niveau micrométrique, présente des avantages avérés et attendus très importants par rapport aux systèmes usuels, et permettra de répondre à de nouveaux besoins.

En effet, miniaturiser, c'est d'abord permettre :

- des gains en poids et volume ainsi qu'en besoin énergétique, ce qui est en particulier très important pour les systèmes embarqués, avec pour corollaires directs :
  - une portabilité du système bien plus grande,
  - une localisation "au plus près" des moyens d'observations, d'actions et de traitements,

- un niveau extrêmement élevé d'intégration de fonctionnalités sur une même puce, avec pour corollaire direct, des performances accrues, puisqu'il est possible de :
  - réduire les temps de transferts et d'échanges des informations entre les différentes fonctions, donc d'augmenter la bande passante et la fréquence des traitements,
  - pouvoir regrouper dans une même puce lesdits moyens d'actionnements, d'observations et de traitements, et donc diminuer le temps de réaction (si important pour les applications temps réel), puisqu'il n'est plus besoin de "remonter" des informations de mesure ou de "redescendre" des informations d'actionnement.

Ces avantages ont encore d'autres corollaires qui sont principalement les suivants.

- L'accroissement du niveau de *sensibilité* et de *sélectivité* à de plus petites ou de nouvelles "variations", grâce à la localisation "micrométrique au plus près", qui permet d'accéder à de nouveaux types d'informations, en rendant possible :
  - d'accéder à de nouveaux *espaces physiques*, comme le corps humain par exemple,
  - d'accéder à un *niveau plus fin de granularité* de grandeurs physiques, notamment des non-linéarités, désormais observables ou commandables, donc, par exemple de nouveaux moyens d'activation tels qu'électrostatique ou électromagnétique,
  - de disposer de capteurs, bien moins intrusifs vis-à-vis de leur environnement.
- Un gain en coût de revient global :
  - les coûts de fabrication des dispositifs, de par les types et quantités de matière première, et aussi de par une production de masse,
  - les coûts de la maintenance et d'un remplacement en cas de défaillance (moins de matériel et d'équipements eux-mêmes réduits).
- Un gain en fiabilité et robustesse lié à :
  - la production de masse, qui permet d'employer des procédés de fabrication éprouvés et qui fournit également des retours d'expérience bien plus significatifs,
  - la suppression des liaisons électriques et des causes de défaillances associées.

5.1.4 Limites

Pourtant, malgré tous ces avantages annoncés, il existe un certain nombre de limitations. Il faut cependant distinguer celles qui sont intrinsèques au concept même de microsystème, de celles liées au manque de maturité des technologies et des connaissances, et qui traduisent en fait les nombreux défis qui restent encore à relever, avec l'espoir donc de lever ce type de limitations. Il faut dire que dans la réalité d'aujourd'hui, la plupart des microsystèmes existants sont développés pour des applications spécifiques, notamment en raison de contraintes d'environnement particulières, ou de la criticité des applications visées.

Les limitations liées au manque de maturité sont : 1) *technologiques*, avec la maîtrise du haut niveau intégration et des technologies de fabrication [Miller *et al.* 1998], de la consommation d'énergie, etc.), 2) *conceptuelles*, avec notamment le besoin d'outils de conception spécifiques, 3) mais aussi *culturelles* et *sociétales*, pour faire accepter ces microsystèmes au grand public.

Mais revenons sur un des défis majeurs qu'est celui de "l'énergie". La miniaturisation a pour avantage de réduire la consommation d'énergie, mais elle a aussi pour inconvénient de réduire la place disponible pour le stockage et les sources de génération de l'énergie, d'où là aussi la nécessité d'autres techniques et technologies. Ce problème d'énergie est d'autant plus crucial lorsqu'il s'agit de systèmes autonomes, voire mobiles (problème de durée d'autonomie). Dans tous les cas, la consommation d'énergie de chaque composant est à évaluer très finement.

Pour ce qui est des limitations intrinsèques, elles sont surtout des contreparties des avantages de la miniaturisation elle-même. Relevons plus particulièrement les limitations suivantes.

- Les *problèmes énergétiques*, qui subsisteront en partie, en dépit des progrès réalisés.
- *L'inaccessibilité* pour des vérifications ou des réparations du fait de leur taille.
- La *fragilité à l'environnement*, de par leur taille, mais aussi, de leur extrême sensibilité aux *microvariations*. Car, l'effet dual de cet avantage est que les microsystèmes sont aussi sensibles à tout un spectre de perturbations, sans effets sur les macrosystèmes, mais parfois très pénalisantes, voire critiques pour les microsystèmes. Il peut s'agir de microdéfauts internes (ex. : les imperfections structurales déjà citées), ou de microperturbations externes, telles que des radiations, notamment dans l'espace.
- *L'inaptitude à agir sur un environnement macroscopique*, par exemple par contact mécanique : un peigne inter-digité électrostatique, mode d'actionnement spécifique aux microsystèmes, génère une force maximale d'à peine quelques nano Newtons.
- *La fiabilité*, qui est une des préoccupations majeures des industriels. Nous illustrons certains de ces aspects, en résumant ici l'exemple présenté dans [Coumar 2003] d'une démarche d'analyse de fiabilité menée chez *Sandia National Lab*, et qui avait pour but de développer un modèle de chaque type de microsystème pour prédire sa fiabilité sans avoir à faire d'essais spécifiques par la suite. Pour crédibiliser les statistiques obtenues, l'identification des mécanismes physiques de défaillance s'appuyait sur la caractérisation d'un grand nombre de composants de chaque type. Le tableau 5.1 résume la classification de S*andia*, en 4 niveaux de fiabilité des produits selon leur complexité et donc le mode de fonctionnement de leurs éléments sensibles : usure mécanique due au mouvement, impacts, frottements, chocs, etc. Moins le produit est complexe, plus il est fiable (ex. : les accéléromètres, peu complexes, sont de classe 1).

	Classe 1	**Classe 2**	**Classe 3**	**Classe 4**
Complexité	- Dispositif immobile	- Parties immobiles - Pas d'impact - Pas de frottement	- Parties mobiles - Impact (s) - Pas de frottement	- Parties mobiles - Impact (s) - Frottement (s)
Exemples	- Accéléromètres - Capteurs de pression - Têtes d'imprimantes	- Gyroscope - Peignes - Filtres	- Relais - Valves - Pompes	- Commutateurs optiques - Serrures
Types de Défaillances	- Contamination particulaire - Collage (dû au choc)	- Idem Classe 1 - Fatigue mécanique	- Idem Classe 2 - Dégâts (impact)	- Idem Classe 3 - Friction, Usure

Tableau 5.1 - Classification des MEMS selon leur fiabilité [Coumar 2003]

Enfin, la principale limitation à long terme de la miniaturisation à l'échelle micrométrique est que l'évolution des technologies atteindra, d'ici à une dizaine d'année, les limites physiques des procédés de réalisation au niveau micrométrique. Et une autre révolution est déjà en cours : celle de la miniaturisation à l'échelle nanométrique, grâce aux nanotechnologies, avec les NEMS, les *Nano Electro Mechanical System.*

En conclusion, la spécificité de la technologie des microsystèmes est d'offrir de meilleures performances à de moindres coûts tout en miniaturisant. Les axes prioritaires pour exploiter les opportunités offertes concernent de multiples applications.

Et une véritable rupture opérationnelle est en vue, par la généralisation de la collecte d'informations et de leurs traitements, avec la notion de réseaux de microsystèmes.

5.2 Nappes et réseaux de microsystèmes

Les nombreux avantages que présentent les microsystèmes à titre individuel, et leur potentiel de progression, conduisent naturellement à vouloir exploiter ces avantages à grande échelle, en cherchant à utiliser des ensembles formés d'un très grand nombre de microsystèmes, au moins sous forme de nappes, si ce n'est de réseaux, c'est-à-dire pouvant communiquer entre eux.

En outre, le principe même de réseau procure des avantages supplémentaires. Ainsi, par exemple, en permettant un plus grand nombre de configurations et de modes dégradés possibles pour les applications s'appuyant sur ces réseaux, la souplesse et la robustesse de ces applications pourraient être accrues. Il devient aussi possible de les utiliser pour des applications distribuées. Mais, ce principe induit de nouvelles problématiques par rapport aux microsystèmes individuels, que l'on peut dissocier en deux grandes catégories :

1) les problèmes de réseau, ou plus généralement de communication,
2) les problèmes des applicatifs s'appuyant sur ces réseaux.

Nous allons d'abord décrire la diversité que recouvre la notion de réseaux de microsystèmes, puis exposer les nouveaux avantages et problématiques génériques, et enfin présenter des exemples dans divers domaines, puis, plus spécifiquement en rapport avec l'aéronautique et notre cas d'étude.

5.2.1 Grands ensembles et classes de microsystèmes

Pour parler de grands ensembles de microsystèmes, le terme *réseau* est souvent utilisé par abus. En réalité, des distinctions plus fines sont à apporter, car un ensemble de microsystèmes n'a d'intérêt que si chacun des microsystèmes a des capacités (donc des fonctionnalités) de communication. Sans quoi, il ne s'agirait que d'un ensemble de microsystèmes mutuellement indépendants, donc sans aucune interaction possible avec un quelconque autre système.

Qui dit interaction, dit architecture et protocole d'interaction, au niveau applicatif et au niveau des communications en support de l'applicatif. La nature et la complexité des interactions dépendront en particulier de la nature (fixe ou mobile) et de l'intelligence de chaque microsystème, mais aussi de leur nombre et de leur hétérogénéité fonctionnelle et structurelle. Tous les cas de figure existent. Cela peut aller d'un ensemble de microsystèmes identiques, fixes, et à gestion totalement centralisée sur un seul calculateur, jusqu'à un ensemble de microsystèmes hétérogènes, mobiles, coopérants et ne communiquant qu'entre eux.

Face à cette diversité, plusieurs classifications ont été faites, et il convient également de donner quelques termes couramment utilisés. Mais, parmi les facteurs de distinction, il en est un qui ne donne pas lieu à une classification particulière, mais qui est quand même fondamental : c'est de distinguer s'il s'agit de communications *entre* les microsystèmes eux-mêmes (intercommunications), ou *avec* les microsystèmes de l'ensemble (extra-communications).

Par la suite, nous utilisons le terme de *réseau* pour décrire le cas de communications complexes (et plus particulièrement entre les microsystèmes), c'est-à-dire s'apparentant aux grands réseaux classiques, et le terme de *nappe* pour décrire le cas de communications plus simples.

Mais d'autres termes sont également utilisés, que l'on peut voir comme des sous-classes des réseaux. On parle de *matrices* de microsystèmes lorsqu'ils sont physiquement disposés et fonctionnellement gérés sous forme matricielle. On parle de *nappes* lorsque qu'il n'y a pas de disposition physique matricielle, et de *nuages* pour des ensembles mobiles.

Cependant, précisons que, dans la suite, par abus de langage et pour simplifier la formulation, nous utiliserons souvent le terme de réseau pour désigner un ensemble de microsystèmes.

Pour les classifications, nous ne reprenons ici que celle de [Collet *et al*. 2004] qui distingue :

1. *Les réseaux de microcapteurs (classe RC) :* ces réseaux permettent au moins la collecte de données sur la zone où ils sont implantés, mais aussi de traiter en partie ces données. Du point de vue du traitement et des communications, cette collecte peut se faire à l'initiative de l'applicatif ou des capteurs (par échantillonnage périodique, ou au besoin). Des exemples d'application sont l'établissement de cartographies (2D ou 3D), la détection ou le suivi de présence pour des applications domotiques ou médicales, ou encore la surveillance du vieillissement ou des efforts structurels (avions, bâtiments).
2. *Les réseaux de microactionneurs (classe RA) :* ces réseaux permettent de distribuer ou de répartir des traitements et des actions sur la zone où ils sont implantés. Des exemples d'application sont les miroirs déformables (formés de plusieurs centaines de micromiroirs et de leurs micropositionneurs électrostatiques), ou les microactionneurs pour modifier les propriétés de la couche limite en aérodynamique.
3. *Les réseaux de microcapteurs et microactionneurs (classe RCA)* qui héritent des deux classes précédentes avec en plus des possibilités d'asservissement locaux.

5.2.2 Avantages et problématiques

5.2.2.1 Avantages

Plusieurs types d'avantages découlent de l'utilisation de réseaux de microsystèmes.

Les traitements applicatifs envisageables sont étendus avec de meilleures performances. Dans les cas les plus avancés, il sera possible de réaliser des traitements distribués. Sans aller jusqu'à là, en restant dans le cas d'une solution centralisée de traitement, il sera possible de généraliser la collecte de données à partir de capteurs, ou, en termes de commandes, d'élaborer des actions macroscopiques par combinaisons d'actions microscopiques.

La fiabilité du système global sera accrue suivant certains aspects. En effet, à contexte équivalent, l'utilisation de plusieurs microsystèmes au lieu d'un macrosystème permet d'introduire, à moindre coût, de la redondance (éventuellement diversifiée), et donc de limiter les conséquences d'une défaillance d'abonnés du réseau. Cette possibilité est d'autant plus intéressante lorsque le niveau d'intégrité de l'information à capter ou à transmettre est important (on ne tolère pas la transmission d'une information erronée). Par exemple, pour un réseau de capteurs couplés à un microprocesseur, celui-ci pourrait calculer la valeur moyenne de toutes les données collectées ou utiliser le processus de vote majoritaire.

Toujours en termes de fiabilité, le niveau d'autorité de chaque microsystème étant plus faible, sa défaillance portera moins à conséquence, et sera davantage tolérable. On peut aussi envisager qu'une défaillance d'un microsystème soit détectée, voire compensée, par un ou des voisins, permettant ainsi, au moins une localisation plus rapide, voire un confinement localisé. Dans le même ordre d'idée, même en fonctionnement normal sans défaillance, les interactions avec des voisins permettraient d'envisager de l'auto-adaptabilité localisée.

Enfin, comme pour les microsystèmes individuels, des gains en coûts d'achat et de maintenance sont attendus. Cependant, d'un autre point de vue, selon la structure physique de la nappe, le remplacement d'un abonné nécessitera en fait le remplacement de toute la nappe. Il est difficile d'évaluer aujourd'hui de quel coté penchera le compromis.

5.2.2.2 Les problématiques

À ces avantages sont inévitablement liés des nouvelles problématiques, spécifiques à l'aspect réseau, et dont certaines sont en fait davantage des perspectives que des limitations.

Avant toute chose, les problématiques de base des communications sont les problématiques classiques des réseaux, c'est-à-dire des problèmes de topologie, d'architecture et de protocoles de communication : routage, identification, diffusion, point à point, etc. Mais les spécificités de microsystèmes nécessitent d'adapter les protocoles connus ou même d'en développer des nouveaux, notamment de par le grand nombre d'abonnés et de leur forte exigence de consommer très peu d'énergie.

Aujourd'hui, les communications dans les réseaux de microsystèmes sont encore très réduites et se résument pratiquement toujours à des liaisons directes vers un calculateur centralisé. Mais ce type de solution devient impossible au-delà d'un certain nombre d'éléments. Or, l'accroissement des communications directes entre les microsystèmes est un aspect capital et inéluctable des grands ensembles de microsystèmes. Cela résulte à la fois des contraintes de robustesse, de programmabilité, d'adaptation et de coopération, de puissance de calcul et de latence de communication [Collet *et al.* 2004], décrits ci-après.

- *Puissance de calcul, latence de communication, adaptation et coopération* : Les contraintes de traitement et de latence de communication dans un réseau de microsystèmes dépendent du nombre et de la localisation de ces microsystèmes, et de la fréquence d'échantillonnage des données échangées. De plus, les réseaux de microsystèmes opérant en temps réel doivent être capables de s'adapter à l'évolution de leur environnement. Le rôle des communications devient dans ce cas critique, à la fois dans la définition des algorithmes d'adaptation et dans l'émergence de mécanismes de coopération entre les microsystèmes pour la réalisation de fonctions globales.
- *Programmabilité* : La répartition et l'ordonnancement des tâches et la synchronisation de leur exécution dans des réseaux de microsystèmes denses et physiquement distribués posent de nombreuses difficultés. D'où la nécessité de créer de nouveaux concepts et algorithmes pour optimiser la distribution des tâches dans ces réseaux.
- *Robustesse* : L'augmentation de la complexité des réseaux de microsystèmes entraîne une fragilisation croissante du fonctionnement.

Donnons maintenant des exemples illustrant l'existant des réseaux et leurs diversités.

5.2.3 Exemples

5.2.3.1 Exemples multidomaines

a) Nappes de géophones pour la détection de vibration (Classe RC)

Les géophones, utilisés dans le domaine de la prospection sismique, sont des capteurs qui transforment en signal électrique des vibrations mécaniques sismiques captées, provoquées par une source, naturelle ou non. Des nappes de microgéophones ultrasensibles, pouvant détecter des vibrations dont les accélérations sont inférieures au millionième de la gravitation terrestre, sont actuellement au stade d'expérimentation pour de la prospection pétrolière sismique de surface, afin d'améliorer les cartographies des couches géologiques. Là aussi une solution centralisée est suffisante, et l'application n'est pas vraiment temps réel.

b) Nappes de capteurs de présence ou de suivi (classe RC)

Le domaine d'applications de ce type de nappes est assez varié. En effet, ils peuvent être utilisés pour la détection de présence ou le confort domotique (ex. : gestion thermique de l'habitat), ou la surveillance médicale (ex. : localisation de personnes pour un suivi des activités, détection de chutes). Pour couvrir une pièce ou une maison, une dizaine de capteurs suffit, avec une fréquence d'échantillonnage de chaque capteur très faible (quelques Hz). Pour une couverture à plus grande échelle (bâtiment, collectif…), le nombre de capteurs à gérer devient plus important (quelques centaines), et nécessite une fréquence de démultiplexage de quelques centaines de Hz. Une solution centralisée de collecte et de traitement sur un seul calculateur reste suffisante du fait du nombre de capteurs.

c) Matrice de micropropulseurs pour micro et nano satellites (Classe RA)

Ce type de satellites, réduit en volume et en poids (quelques dizaines de kg, ou moins) nécessite un système de propulsion réduit (de 1 μN à 10 N). Un module de micropropulsion a été assemblé au LAAS-CNRS [Rossi *et al.* 1999]. Il comprend la matrice de micropropulseurs et le système électronique dont l'architecture consiste en plusieurs blocs connectés à un bus : une source d'énergie programmable, un circuit de mesure de température, des mémoires, un multiplexeur à l'entrée de la matrice de micropropulseurs à commander. Là encore, une solution centralisée est suffisante, mais cette fois, il s'agit d'une application temps réel.

d) Matrice de microéjecteurs pour la synthèse in situ d'ADN sur bio-puces (classe RA)

Il s'agit de réaliser des têtes d'éjection pour déposer de manière très localisée et à une fréquence de l'ordre du Hz des gouttes de quelques nano litres (voire moins) [Gue *et al.* 2000]. Une tête comporte de 100 à 10000 éjecteurs disposés en matrices et adressables individuellement pour réaliser les séquences d'ADN désirées.

e) Nappes de microinterrupteurs pour circuits reconfigurables (classe RA)

Ce type de réseaux temps réel peut s'appliquer pour les communications par satellites et pour la téléphonie mobile. Il s'agit de nappes de têtes d'émission-réception très compactes, à faible coût, et présentant des performances en termes de linéarité, de pertes d'insertion, d'isolation et de consommation qui sont nettement supérieures à celles des solutions conventionnelles.

5.2.3.2 Exemples dans le domaine aéronautique

Toujours sans être exhaustif, attachons nous maintenant plus en détail à quelques exemples plus spécifiques à notre domaine d'application : l'aéronautique.

a) Réseaux de capteurs de contraintes pour la surveillance du vieillissement de la structure d'un avion

Le but de ces réseaux de capteurs est de mesurer puis d'enregistrer les données de vibrations dans les points névralgiques d'une structure mécanique d'un avion [Yang & Han 2002]. Chaque capteur peut être interrogé par le calculateur central, ou l'alerter, selon des critères de contraintes maximales, pour signaler un dysfonctionnement. Il se met en "sommeil" lorsque les conditions le permettent et se réactive automatiquement grâce à un système de veille embarqué. Le calculateur central collecte les données de tous les capteurs pour nourrir une base de données qui permet des analyses plus précises et du diagnostic. C'est encore un exemple de solution centralisée et non temps réel.

b) Nappes de microcapteurs thermiques pour la détection de feu dans un réacteur

Un réacteur d'avion est une niche très riche pour l'utilisation de réseaux de microsystèmes. La DARPA mène des études sur l'implantation de nappes de microcapteurs acoustiques, de vibration et de pression. Mais détaillons ici le cas de nappes pour la détection de feu. Il s'agit de répartir dans un réacteur 10^4 microcapteurs thermiques, qui sont des microtiges en nickel de 150 μm de diamètre et de 500 μm de longueur et espacés de 1 mm. Le traitement est centralisé sur un calculateur qui multiplexe les informations émises par les microcapteurs à une fréquence de l'ordre de 1 Mhz, pour satisfaire une latence maximale admissible de réaction de 1 seconde.

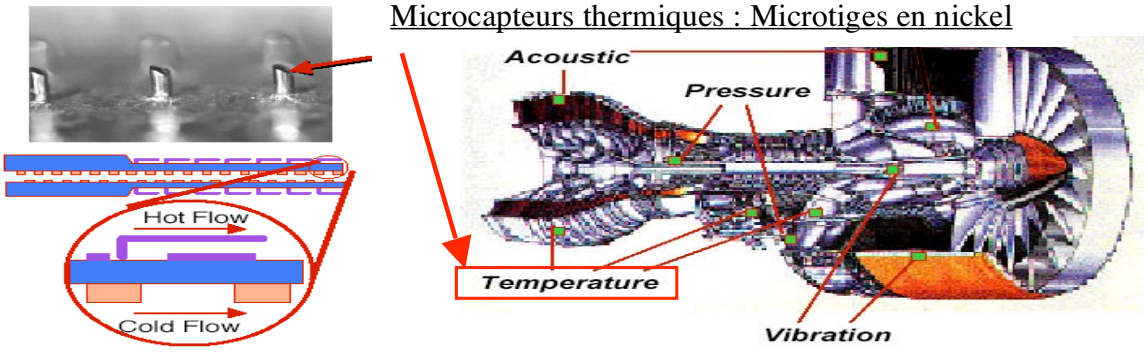


Figure 5.5 - Nappes de microcapteurs thermiques

c) Réseaux de capteurs de pression répartis sur une aile d'avion

Le but est de faciliter et d'améliorer la vérification de la charge exercée sur les ailes ou le fuselage, en vol et au sol [Kim et al. 2001]. Il s'agit d'une perspective envisagée par Boeing. Les microcapteurs de pression sont assemblés dans de multiples modules multi-puces formant des "ceintures" de capteurs de pression (cf. figure 5.6). Ces modules ont des fonctionnalités variées comme le traitement des données émises par l'ensemble des microcapteurs, ou aussi l'auto-calibration [Eccles *et al.* 2001]. Et ces modules peuvent communiquer entre eux et fournir au calculateur central des informations groupées sur la pression.

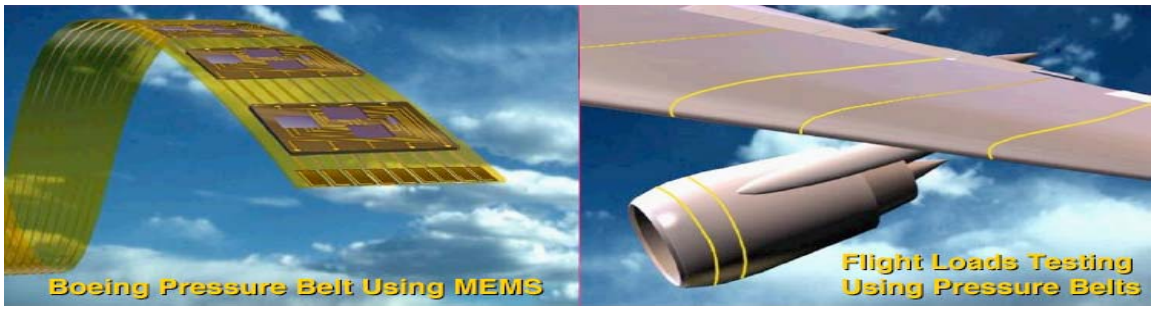


Figure 5.6 - Réseaux de capteurs de pression répartis sur une aile d'avion

d) Le projet AEROMEMS

Ce projet a pour but de démontrer que les microsystèmes peuvent être utilisés pour commander les écoulements des flux d'air (laminaires et turbulents) [Kumar *et al.* 1998] par rapport aux couches limites en dessous des ailes, améliorant ainsi la manœuvrabilité de l'avion et surtout sa pénétration dans l'air en réduisant considérablement les forces de frottement sur le fuselage et par la suite la consommation de carburant [Katnet 1997]. Des études théoriques et des analyses numériques ont été entreprises pour comprendre et mesurer les paramètres d'écoulement exigés pour développer des capteurs, des actionneurs et des stratégies de commande [Katnet 2002].

e) Le projet AWIATOR

Ce projet européen AWIATOR (Aircraft Wing with Advanced Technology Operation) vise à valider des technologies avancées en matière de conception d'ailes pour les futurs avions (http://www.awiator.net). Une des idées est de développer des ailes "adaptatives" en forme et volume, pour mieux ajuster le comportement de l'avion selon les variations des charges sur les ailes entre les différentes phases de vol [Hansen 2003]. Les objectifs visés sont notamment une diminution de 5 à 7 % des frottements sur les ailes, notamment au décollage et à l'atterrissage, ainsi qu'une accélération de la destruction des tourbillons de sillage [Goular *et al.* 2004].

À terme, il est prévu d'ajouter une dizaine de mini-surfaces mobiles sur chaque bord de fuite des ailes pour optimiser le contrôle de la portance (cf. figure 5.7). Pour les moyens d'activation des surfaces, il est prévu de passer des dizaines de vérins hydrauliques utilisés aujourd'hui à des nappes de quelques dizaines ou centaines de mini-vérins hydrauliques qui activeront les mini-surfaces mobiles ajoutées pour former les ailes adaptatives.

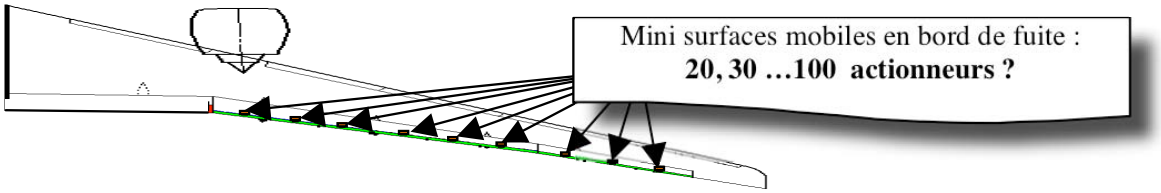


Figure 5.7 - Projet AWIATOR : mini-surfaces additionnelles

5.2.4 Notre champ d'intérêt

Les deux projets AWIATOR et AEROMEMS traitent en partie des possibilités et des avantages d'utiliser des réseaux de microsurfaces de contrôle dans le système de commandes de vol d'un avion, étant rappelé que le rôle et les constituants des CDV ont été exposés dans le chapitre 1. Ces aspects, précisés ci-dessous, constituent notre cas d'étude pour nos travaux.

5.2.4.1 Réseaux de microsurfaces de contrôle

Les tailles envisagées pour de telles microsurfaces de contrôle sont de l'ordre du mm^2, cm^2 voire dm^2. Il est prévu qu'elles viennent en remplacement ou en complément des surfaces actuelles pour obtenir de meilleurs niveaux de manœuvrabilité de l'avion avec une consommation d'énergie réduite [Ho *et al.* 2003], et une meilleure robustesse.

En termes de manœuvrabilité, leur apport vient du fait qu'elles permettront de mieux contrôler l'écoulement de l'air sur les ailes. Des études réalisées sur des drones ont montré qu'en commandant de façon optimale des nappes de microflaps disposées sur les ailes d'un drone, il est possible d'accroître les actions en moyenne de 31% sur la portance de l'avion et de 17% sur sa poussée [Ho & Tai 1996]. En effet, de par la taille de ces nappes :

- elles peuvent être placées à des emplacements stratégiques de l'aile, qui ne sont pas encore exploitables aujourd'hui (cf. figure 5.8),
- une microsurface de contrôle permet de générer une force aérodynamique ponctuelle qui, additionnée à celles produites par les autres, produit des forces et des moments différents de ceux produits aujourd'hui, avec de surcroît, un niveau d'action plus fin.

Par ailleurs, le poids, l'encombrement et les besoins énergétiques influent beaucoup la conception d'un avion. Réduire ces paramètres est une préoccupation constante des constructeurs pour abaisser les frais d'exploitation des compagnies aériennes.

Or, l'utilisation des microsurfaces de contrôle nécessitera des moyens d'activations réduits et plus légers, ce qui permettra ou imposera même de faire appel à une autre forme d'énergie d'activation, qui sera une autre source de gain en poids, et donc de consommation.

Car actuellement, les moyens d'activation s'appuient sur une technologie de type hydraulique (des vérins hydrauliques), et la taille des surfaces de contrôle fait que leur inertie est parfois très élevée et exige des systèmes d'activation puissants, donc gourmands en énergie, mais aussi lourds et volumineux [Huang 2001]. Par exemple, dans le cas de l'Airbus A380, le gouvernail de direction mesure environ 23 mètres.

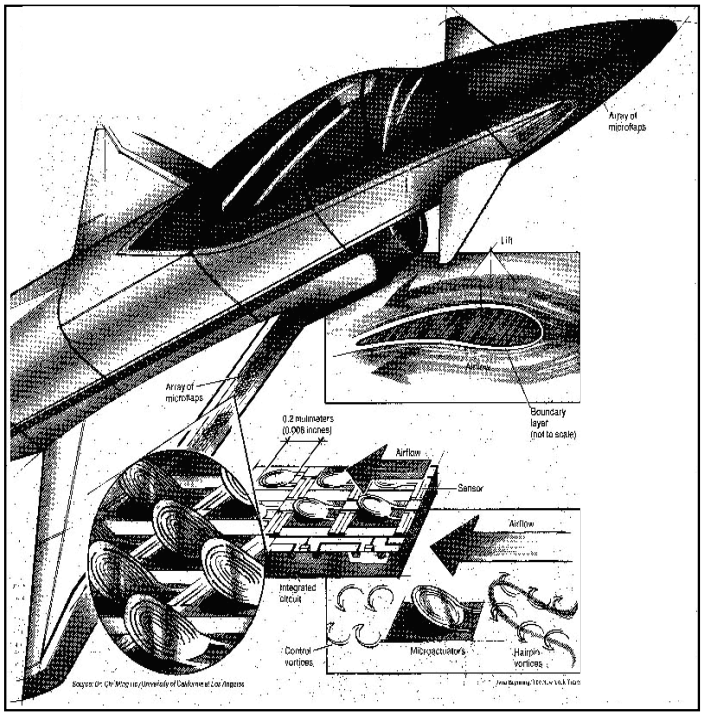


Figure 5.8 - Nappes de *microflaps* sur les ailes d'un avion (http://www.darpa.mil)

Or, pour des microsurfaces, il est presque certain qu'en dessous d'une certaine taille (de l'ordre du mm^2), il ne sera plus possible de les activer hydrauliquement, car le facteur de réduction des vérins a une limite technologique. Il faut alors envisager d'autres techniques d'activation des microsurfaces de plus petite taille [Ho *et al.* 1994] : électromagnétique, électrostatique, thermomécanique, piézoélectrique, électrodynamique, etc. [Liu *et al.* 1995, Grosjean *et al.* 1998]. Les plus étudiées étant les techniques électromagnétiques [Judy & Muller 1997] et électrostatiques [Kimura *et al.* 1997] (cf. figure 5.9).

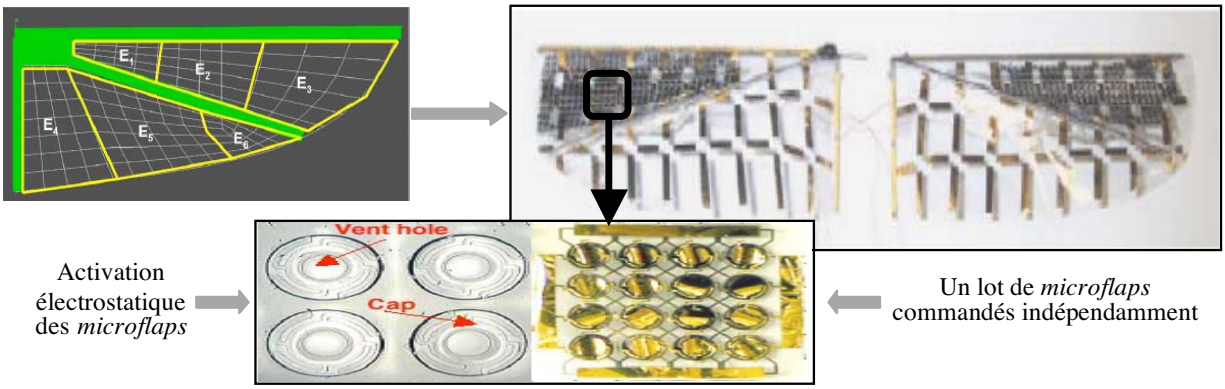


Figure 5.9 - Activation électrostatique de nappes de microflaps

Enfin, un autre apport important attendu de l'utilisation des microsurfaces de contrôle est une robustesse accrue des CVF qui seront davantage tolérant aux défaillances, de par le nombre élevé des microsurfaces utilisées. En perdre quelques dizaines ou centaines peut ne pas être considéré comme un événement redouté, alors qu'actuellement, un tel événement correspond à l'embarquement de deux spoilers sur un Airbus A320 par exemple.

5.2.4.2 Nos considérations et hypothèses dans ce contexte

Ce contexte des réseaux de surfaces de contrôle a été le cadre d'application de nos travaux sur la définition d'un système de communication en vue de commander des nappes de quelques centaines ou milliers de microactionneurs. Or, comme tout domaine investi par des microsystèmes, ce contexte soulève de nombreuses problématiques, déjà évoquées d'un point de vue général, et qui se déclinent maintenant par rapport aux spécificités du domaine.

Il convient donc de préciser, parmi toutes ces problématiques, celles que nous avons abordées, sachant que de plus, du fait que ce contexte est très novateur, certaines problématiques sont accrues. En effet, concrètement, les microsystèmes envisagés ne sont pas du tout à maturité et ne sont même pas encore à disposition. Il subsiste donc des inconnues, dont en particulier le facteur d'échelle de réduction possible ou souhaité pour ces surfaces (1/1000, 1/10000, 1/100000), avec comme inconnue induite, les problèmes relatifs à leur moyen d'activation : conserver l'activation hydraulique ou opter pour d'autres techniques d'activation ?

Nous n'avons donc raisonné que sur le principe de l'existence de ces microsurfaces, et non sur leurs réelles caractéristiques techniques quantifiées. Et nous n'avons envisagé qu'un seul type de microsurfaces, des microspoilers, qui sont des surfaces non critiques. Et enfin, nous n'avons retenu comme critères d'impact que le nombre élevé de microsystèmes, invariant en nombre et localisation, et peu complexes dans le sens où ils ne présentent pas, ou peu de capacités de mémorisation ou de traitement local, et surtout pas de capacités d'intercommunication [DeLabachelerie *et al.* 2001]. D'où notre utilisation du terme de *nappes* plutôt que de réseaux.

Enfin, rappelons que nous nous positionnons en qualité *d'utilisateurs* de microsystèmes. Nous n'avons donc pas traité les problématiques de conception, de génération et de consommation d'énergie, de technologies de fabrication ou d'activation des microsystèmes [Lyshevski 2002].

5.3 Système de communication en vue de la commande de nappes de microactionneurs : cas des CVF

Les bases architecturales et protocolaires du système de communication envisagé ont été présentées au chapitre 2. Mais de nombreux et importants points à résoudre avaient dû être mis en attente. Nous développons maintenant ici plus particulièrement les aspects architecturaux de ce réseau, pour ensuite présenter des pistes pour la répartition des traitements applicatifs et des pistes sur la "répartition" des aspects sûreté de fonctionnement, dont notamment les besoins ou les possibilités de moduler le niveau de cette sûreté de fonctionnement. Des premiers éléments de validation de cette architecture seront donnés, en considérant l'utilisation de protocoles de types synchrones. Le tout est sous-tendu par les CVF. Mais avant cela, nous rappelons les principales contraintes que doit satisfaire le système de communication.

5.3.1 Rappels des contraintes à satisfaire

Dans les SCC considérés, les asservissements (génération de commandes à partir de retours capteurs) gérés par le calculateur doivent être rafraîchis avec une période généralement très faible, que nous avons prise égale à 10 ms. Le temps de traitement Tt des données par le calculateur doit être inférieur à la plus petite période d'échantillonnage Te des différents capteurs. Le grand nombre d'abonnés est également un facteur pénalisant. Il faut aussi prendre en compte la latence ΔTc de communication et le temps de réponse des différents capteurs (mais généralement, ce temps de réponse est négligeable devant ΔTc).

ΔTc dépend en général du protocole de communication utilisé et de la localisation dans le réseau des données sur lesquels se base le traitement. Nous allons donc considérer comme contrainte temporelle à respecter : $Tt + \Delta Tc < Te$.

La robustesse du système de communication est en partie prédéterminée par les choix retenus en matière d'architectures physiques, de communications et de traitements. Or, des choix parfaitement adaptés pour satisfaire des contraintes temps réel peuvent s'avérer peu efficaces pour ce qui est du maintien des fonctionnalités globales du système en présence de défaillances, au niveau physique, réseau ou application. En effet, pour ce qui est des contraintes d'intégrité sur les données transmises de bout en bout du système de communication, elles sont considérées suivant trois niveaux [Collet *et al.* 2004] :

- le niveau physique, qui représente les supports physiques,
- le niveau réseau, qui définit les modes de communications et les protocoles associés,
- le niveau application ou traitement, qui spécifie l'architecture générale du système par rapport à la répartition du traitement.

5.3.2 Architecture physique

5.3.2.1 Architecture arborescente

Comme dit au chapitre 2 (cf. § 2.1.1), le manque de maturité des réseaux de microsystèmes fait, qu'à ce jour, on constate que la plupart de ces réseaux sont bâtis autour d'architectures centralisées. Mais ce type d'architecture n'est plus adapté au-delà d'un certain nombre de microsystèmes, le calculateur ne pouvant plus à lui seul assurer toute la charge de la gestion des communications et des traitements applicatifs. Cela amène à considérer une architecture complexe de communication, avec des nœuds intermédiaires, actifs pour une part d'entre eux.

Compte tenu que nous visons des SCC, une architecture qui paraît appropriée est une architecture arborescente, à base de bus interconnectés, avec plusieurs niveaux de nœuds intermédiaires entre le calculateur et les nappes de microactionneurs (cf. § 2.1.1.4).

Pour faciliter la présentation qui suit, nous rappelons, sur la figure 5.10, le principe simplifié et générique de l'architecture physique que nous avons retenue et déjà montrée sur la figure 2.1.

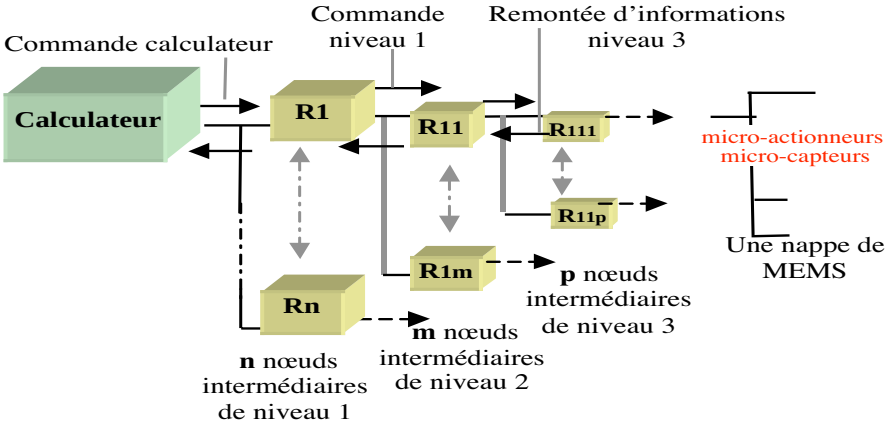


Figure 5.10 - Architecture physique en arbre du système de communication

Dans cette architecture, les fonctionnalités des nœuds intermédiaires peuvent être différentes selon : 1) leur niveau dans l'arborescence et donc le nombre des nœuds dont ils sont en charge, 2) la nature du traitement qu'ils auraient à réaliser sur la commande.

En effet, la commande émise par le calculateur aux nœuds intermédiaires auxquels il est directement connecté (nœuds intermédiaires de premier niveau) peut être mémorisée, voire traitée par ces nœuds suivant la nature et la valeur de l'information remontée par d'autres nœuds intermédiaires.

La commande est ensuite envoyée à un ou des nœuds intermédiaires de niveau inférieur et ainsi de suite jusqu'à ce que la commande initiale du calculateur arrive aux nappes de microactionneurs.

5.3.2.2 Architecture physique dans le cas des CVF

Appliquons maintenant le principe de cette architecture en arbre aux CVF, en raisonnant sur des surfaces de contrôle de type spoilers, qui agissent sur le roulis de l'avion (cf. § 1.5.2.2.1).

Le nombre de microspoilers dépend de leur taille, qui dépend elle même en partie de la technologie d'activation (cf. § 5.2.4.1). Prenons comme hypothèse, un facteur de réduction de la taille d'un spoiler de 1/1000. Dans ce cas, en prenant comme exemple l'Airbus A320, les 10 spoilers d'aujourd'hui, assurant chacun 10% du contrôle en roulis de l'avion, seraient remplacés par 1000 microspoilers assurant chacun à 0,01% de l'autorité de contrôle.

Alors, on obtient une architecture en arbre à 4 niveaux avec 10 nœuds intermédiaires par niveau, telle que celle montrée par la figure 5.11.

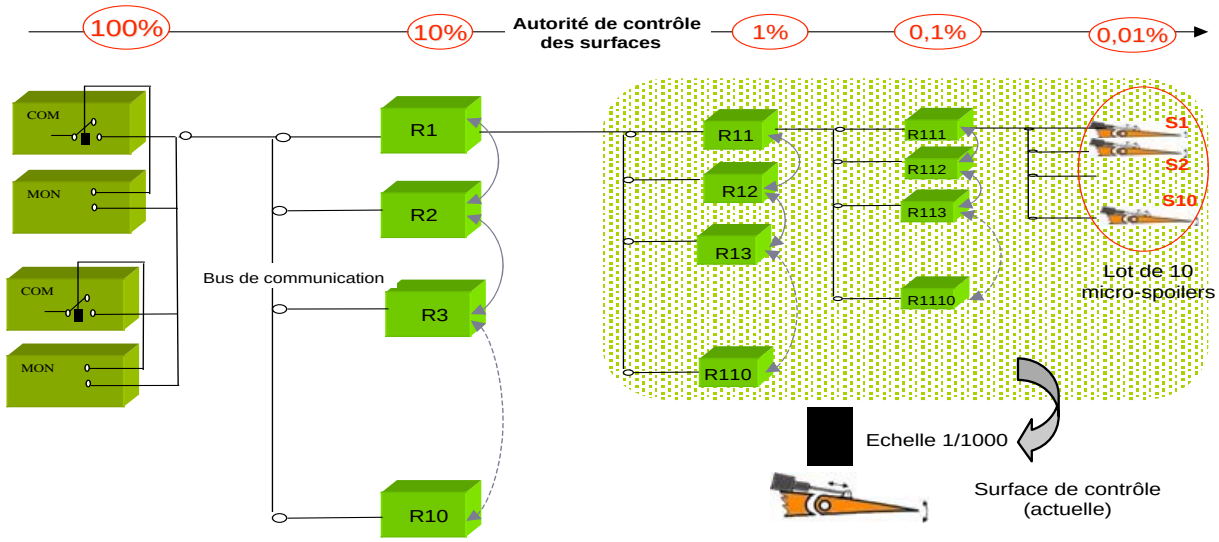


Figure 5.11 - Architecture physique pour commander des nappes de microspoilers

Pour chacun des nœuds intermédiaires du premier niveau de l'architecture, l'autorité de contrôle en roulis de l'avion est de 10%, ce qui correspond à l'autorité de contrôle d'un spoiler actuel. Puis au second niveau, chaque nœud intermédiaire aurait une autorité de contrôle de 1% et ainsi de suite jusqu'au dernier niveau qui représente un grand nombre de microspoilers, qui pourraient éventuellement n'être commandés que selon deux états (actifs, inactifs), et dont chacun aurait une autorité de contrôle de 0,01%.

5.3.3 Architecture de traitement pour la commande de nappes de microactionneurs

Sur la base de l'architecture physique qui vient d'être présentée, nous allons maintenant présenter des pistes pour l'architecture de traitement qui pourrait être appliquée. L'architecture de traitement décrit l'organisation du système pour mettre en œuvre les fonctionnalités décrites par des algorithmes de traitement. Le but de cette architecture est de répondre à la problématique suivante : "Comment assurer la répartition des traitements relatifs aux commandes et retours de positions (émis respectivement par les calculateurs et les nappes de microsystèmes) entre les différents niveaux de nœuds intermédiaires, pour commander et asservir les nappes de microactionneurs tout en respectant les contraintes temps réel ?"

En pratique, la définition de cette architecture obéit à des règles qui découlent (au moins dans un premier temps) de la structuration du réseau de communication et de la spécificité de chaque algorithme de traitement. Les questions critiques à ce niveau sont :

- Où sont effectués les traitements des données dans le système de communication ?
- Quelle fraction du traitement est distribuée ?
- Comment sont liés les traitements et les communications ?

Le système de communication pour commander les nappes de microactionneurs peut être vu de manière hiérarchisée, en distinguant plusieurs couches d'abstraction qui vont se répercuter sur la manière de voir la commande. Avec cette vision, il apparaît au moins une couche : les "lots de microcapteurs". Dans ce cadre, nous proposons un contrôle distribué selon trois couches :

- La première couche est relative aux nœuds intermédiaires appartenant aux niveaux supérieurs de l'architecture du système de communication, présentée à la figure 5.11. Cette couche a une vision globale de l'ensemble des lots de microactionneurs. Elle définit les tâches à transmettre aux couches inférieures au travers de : l'adaptation, la lecture des microcapteurs, l'évolution, la reconfiguration, l'analyse, la coordination, l'organisation, la décomposition, la prise de décision sur la base d'analyses de performances, de prédiction du comportement du système et de diagnostic.
- La deuxième couche est relative aux nœuds intermédiaires des niveaux inférieurs de l'architecture, et contrôle des lots de microactionneurs. Elle traite des commandes pour la couche en dessous, pour coordonner les actions entre les différents microspoilers.
- La dernière couche gère les microactionneurs au niveau individuel, en appliquant un signal de commande à chacun. Les services fournis par cette couche, dans le cas des systèmes de CDV, et notamment des nappes de microspoilers, peuvent être :
  - *Déplacement* : ce service déplace la surface du microspoiler pour influer sur les mouvements de l'avion au tour de son axe de roulis ;
  - *Asservissement* : ce service élimine l'erreur éventuelle entre la consigne et la position (ou déplacement) réelle de la microsurface, ce qui nécessite un capteur pour connaître cette position réelle et la comparer à la consigne ;
  - *Diagnostic et détection d'erreurs* : par exemple, un microspoiler à activation électromagnétique pourrait être pourvu, en plus d'un capteur de position, d'un capteur d'intensité du champ électromagnétique autour de la microsurface, pour détecter une anomalie non perçue par le capteur de position.

L'architecture de traitement entre le dernier niveau de nœuds intermédiaires et les nappes de microactionneurs est plutôt centralisée pour respecter l'hypothèse de travail considérant que les microsystèmes sont dépourvus de capacités d'intercommunication.

Paradoxalement, en termes de sûreté de fonctionnement, les nœuds intermédiaires peuvent apporter des modifications de plus en plus grandes sur les commandes reçues à chaque passage à un niveaux inférieur, car une défaillance du traitement applicatif d'un nœud intermédiaire aura moins de conséquence quand le nombre de microactionneurs concernés par sa commande est faible (figure 5.12). Donc, aux niveaux inférieurs, l'intégrité des communications, et plus particulièrement le pouvoir de détection des altérations des messages transmis sera moins recherchée que dans les premiers niveaux intermédiaires. Ce pouvoir de détection d'erreurs peut être presque négligeable au dernier niveau de l'architecture : les nœuds intermédiaires n'y commandent que quelques dizaines de microsystèmes pouvant être sans capacités de traitement local et de mémorisation. Dans ce cas, ils peuvent n'avoir que deux états : actif, inactif.

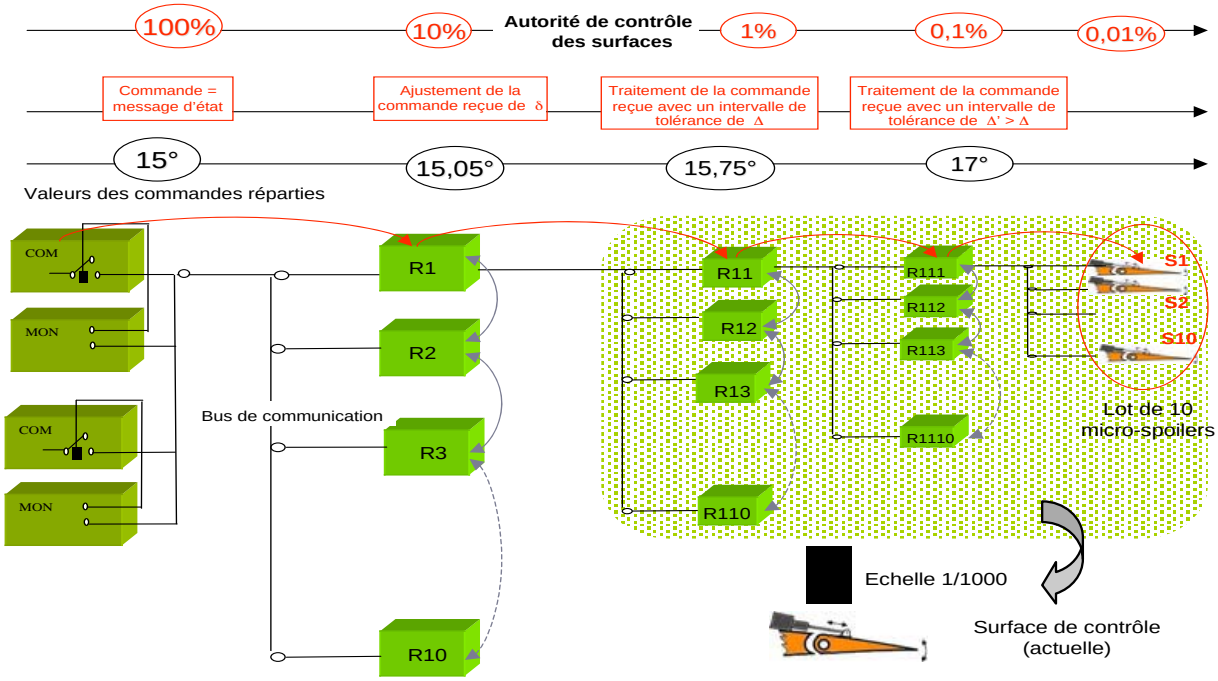


Figure 5.12 - Exemple de répartition de traitement dans le système de communication en vue de commander des nappes de microspoilers

5.3.4 Architecture de communication en vue de la commande des nappes de microspoilers

Présentons maintenant l'architecture et le protocole de communication en vue de la commande des nappes de microspoilers. Un exemple d'architecture pour répartir le traitement réseau entre les nœuds du système de communication pourrait être celui présenté ci-après.

Les fonctionnalités réseau des nœuds intermédiaires sont de moins en moins complexes à chaque passage vers des niveaux inférieurs de nœuds intermédiaires, vu qu'à chacun de ces passages, le nombre de microspoilers concernés par la commande du calculateur diminue. En effet, les nœuds intermédiaires de premier niveau peuvent avoir des fonctionnalités réseau qui représenteraient une capacité de mémorisation et éventuellement de traitement local des données. Par contre, les nœuds intermédiaires de niveau inférieur n'ont pas besoin d'avoir des fonctionnalités réseau importantes, vu le petit nombre de microspoilers dont ils sont en charge.

Cette architecture de communication devra assurer une robustesse du système de communication, notamment par rapport à l'intégrité des données sont transmises, à travers les différents niveaux de nœuds intermédiaires.

Les problématiques relatives à l'intégrité des données échangées et qui doivent être résolues par l'architecture de communication sont :

- Jusqu'à quel niveau du réseau doit-on assurer un haut niveau d'intégrité sans nuire aux contraintes temps réel du système ?
- Quels types d'intégrité des communications doit-on assurer dans les niveaux inférieurs de l'architecture et avec quels moyens ?

Pour répondre à ces problématiques, commençons par préciser les besoins du système de communication en termes d'intégrité, suivant les différents niveaux de l'architecture.

Ces besoins se basent sur le fait que, plus grand est le nombre de microspoilers reliés à un nœud de cette architecture, soit directement, soit via plusieurs niveaux de nœuds intermédiaires, plus il faut assurer un haut niveau d'intégrité des données échangées entre le calculateur et le nœud en question. Par contre, plus on passe à un niveau inférieur de l'architecture, plus le nombre de microspoilers concernés par la commande diminue, et donc moins le niveau d'intégrité à assurer pour les communications sera important. Cette réduction de niveau d'intégrité implique généralement un gain en termes de temps de calcul des données de contrôle, de temps de latence, de débit utile et donc de bande passante.

Un exemple de répartition des moyens assurant l'intégrité des données transmises entre les différents niveaux de l'architecture de nœuds intermédiaires pourrait être le suivant :

- 1er niveau : Utilisation d'une fonction de contrôle d'erreur évolutive (par ex., CRC évolutif), dont nous rappelons que l'apport en termes d'intégrité, par rapport à des fonctions de contrôle d'erreur classiques, a été validée par rapport à des contraintes d'intégrité et de temps réel pour des systèmes particuliers considérés par notre étude.
- 2ème niveau : Utilisation de la mise à jour incrémentale relative à un code de contrôle d'erreur linéaire (ex. : mise à jour incrémentale du CRC). Cette technique s'adapte plus particulièrement dans le cas où le nœud intermédiaire aurait des fonctionnalités de traitement des données reçues.
- 3ème niveau : Utilisation d'un code détecteur d'erreurs classique (ex. : CRC).
- 4ème niveau : Utilisation d'un code détecteur d'erreurs plus simple (ex. : contrôle par somme).
- 5ème niveau : Utilisation d'un code détecteur d'erreurs encore plus simple (ex. : code de parité).
- 6ème niveau : Aucun moyen de détection d'erreurs.

5.3.5 Premiers éléments de validation des choix architecturaux et protocolaires par rapport aux contraintes temps réel

Le but de cette section est de donner les premiers éléments de la validation de nos choix architecturaux et protocolaires pour les CVF, par rapport aux contraintes temps réel, en rappelant que, pour ce qui est des protocoles, nous avons conclu que pour les SCC considérés, des protocoles de type synchrone étaient à privilégier (cf. § 2.1.2).

Une telle validation devra en réalité répondre à la problématique suivante : "À partir de combien de niveaux de nœuds intermédiaires n'est-il plus possible d'assurer une commande des nappes de microspoilers satisfaisant aux contraintes temps réel ?".

Dit autrement, le but est de vérifier si la latence de transmission de la commande du calculateur aux nappes de microspoilers, via le système de communication présentant plusieurs niveaux de nœuds intermédiaires, satisfait les contraintes d'exécution temps réel et qui imposent un rafraîchissement de la commande adressée à chaque microspoiler toutes les 10 ms. Le raisonnement est identique pour vérifier la latence des transmissions des retours de positions émises par les microcapteurs.

Cette vérification de la latence de transmission de la commande représente en réalité une extrapolation des études présentées dans [Youssef *et al.* 2003b], dans le cadre d'une première approche du problème de définition d'un système de communication pouvant commander des nappes de microspoilers. Dans le document cité, nous avions représenté le cycle de base d'un système de communication simplifié, ayant un seul niveau de nœuds intermédiaires, s'appliquant aux surfaces de contrôle actuelles des systèmes de CDV (cf. § 2.2.2).

Le tableau 5.2 et la figure 5.13 illustrent un exemple d'ordonnancement statique des fenêtres temporelles autorisant l'accès en écriture au bus de communication des nœuds intermédiaires reliant le calculateur au 1er lot de microspoilers.

La figure 5.13 montre également le cycle global du système de communication qui est composé de quatre cycles de base, chacun relatif à un niveau de l'architecture physique en arbre pour la commande de nappes de microspoilers présentée au § 5.3.2.2.

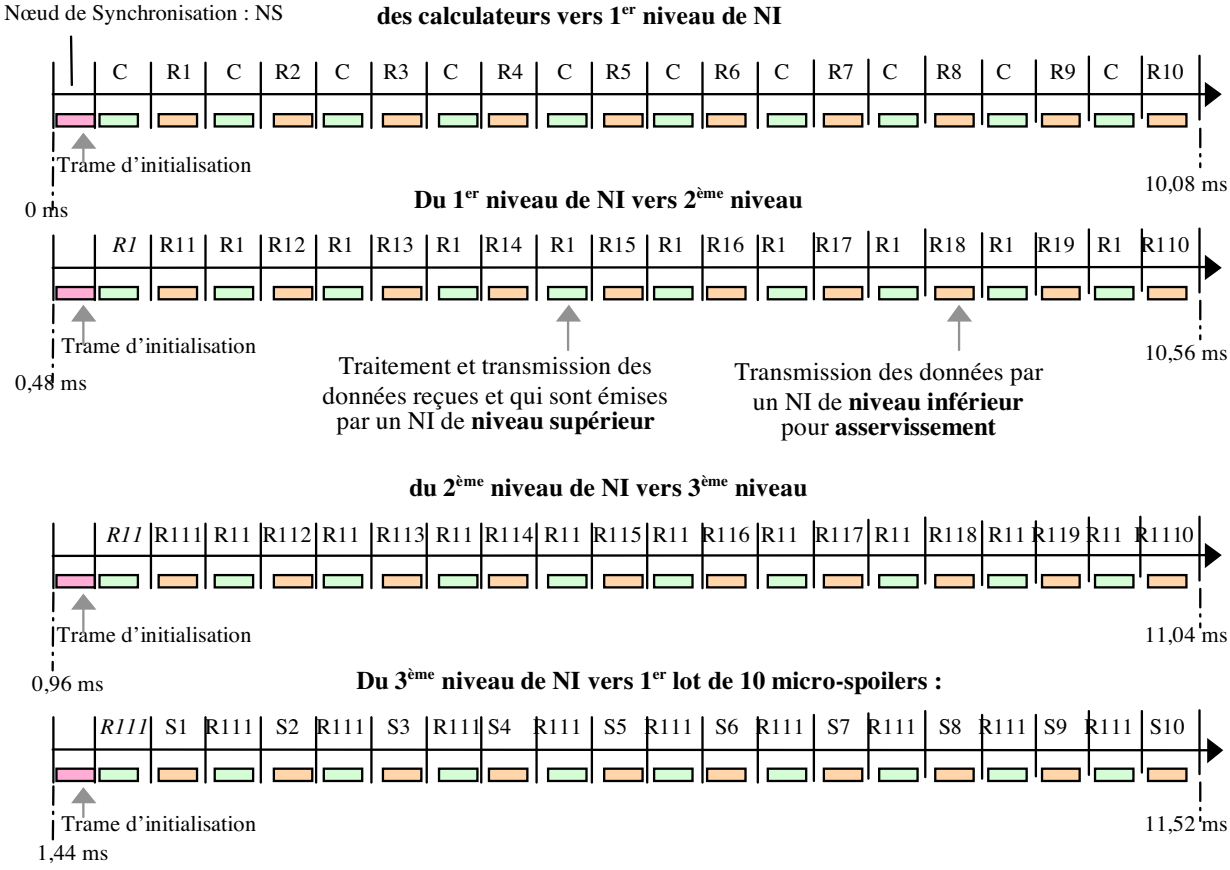


Figure 5.13 - Cycle global du système de communication (basé sur les bus reliant le calculateur au 1er lot de microspoilers)

Transmission de la commande du calculateur au **1er lot de micro-spoilers**

Calculateurs → 1er niveau de NI

Nœud	NS	C	R1	C	R2	C	R3	C	R4	C	R5	C	R6	C	R7	C	R8	C	R9	C	R10
Fenêtre temporelle (ms)	0	**0.48**	0,96	1,44	1,92	2,4	2,88	3,36	3,84	4,32	4,8	5,28	5,76	6,24	6,72	7,2	7,68	8,16	8,64	9,12	9,6
	0.48	**0.96**	1,44	1,92	2,4	2,88	3,36	3,84	4,32	4,8	5,28	5,76	6,24	6,72	7,2	7,68	8,16	8,64	9,12	9,6	10,08

1er niveau de NI → 2ème niveau de NI

Nœud	NS	R1	R11	R1	R11	R1	R11	R1	R11	R1	R11	R1	R11	R1	R11	R1	R11	R1	R11	R1	R11
Fenêtre temporelle (ms)	0.48	0,96	1,44	1,92	2,4	2,88	3,36	3,84	4,32	4,8	5,28	5,76	6,24	6,72	7,2	7,68	8,16	8,64	9,12	9,6	10,08
	0.96	1,44	1,92	2,4	2,88	3,36	3,84	4,32	4,8	5,28	5,76	6,24	6,72	7,2	7,68	8,16	8,64	9,12	9,6	10,08	10,56

2ème niveau de NI → 3ème niveau de NI

Nœud	NS	R11	R111	R11	R111	R11	R111	R11	R111	R11	R111	R11	R111	R11	R111	R11	R111	R11	R111	R11	R111
Fenêtre temporelle (ms)	0,96	1,44	1,92	2,4	2,88	3,36	3,84	4,32	4,8	5,28	5,76	6,24	6,72	7,2	7,68	8,16	8,64	9,12	9,6	10,08	10,56
	1,44	1,92	2,4	2,88	3,36	3,84	4,32	4,8	5,28	5,76	6,24	6,72	7,2	7,68	8,16	8,64	9,12	9,6	10,08	10,56	11,04

3ème niveau de NI → 1er lot de 10 microspoilers : S1… S10

Nœud	NS	**R111**	S1	R111	S2	R111	S3	R111	S4	R111	S5	R111	S6	R111	S7	R111	S8	R111	S9	**R111**	S10
Fenêtre temporelle (ms)	1,44	**1,92**	2,4	2,88	3,36	3,84	4,32	4,8	5,28	5,76	6,24	6,72	7,2	7,68	8,16	8,64	9,12	9,6	10,08	**10,56**	11,04
	1,92	**2,4**	2,88	3,36	3,84	4,32	4,8	5,28	5,76	6,24	6,72	7,2	7,68	8,16	8,64	9,12	9,6	10,08	10,56	**11,04**	11,52

Réception de la commande du calculateur (après son traitement par les nœuds intermédiaires) par le **microspoiler S1**

Réception de la commande du calculateur (après son traitement par les nœuds intermédiaires) par le **microspoiler S10**

Tableau 5.2 - Illustration de l'ordonnancement statique des fenêtres d'émission

Chaque cycle de base de ce système s'étend sur 10 ms. Avec la contrainte temporelle de rafraîchissement des microspoilers et en raisonnant par rapport à un seul bus, nous avons 20 asservissements par bus à garantir dans un délai de 10 ms.

À raison d'une commande et d'un retour par asservissement, nous avons donc 20 messages échangés, sur chaque bus et toutes les 10 ms, entre un nœud intermédiaire et les dix autres nœuds de niveau inférieur qu'il doit commander. Au premier cycle de base, les messages échangés le sont entre le calculateur et les 10 nœuds intermédiaires de premier niveau.

À ces 20 messages échangés par bus, il faut ajouter le message d'initialisation diffusé aux différents nœuds connectés au bus, au début de chaque cycle de base. Soit au total 21 messages échangés sur chaque bus durant les 10 ms, ce qui correspond à 21 fenêtres temporelles (cf. figure 5.13). Avec une allocation équi-répartie des fenêtres temporelles, la largeur de chaque fenêtre est de 10/21 ms, soit 0,48ms, qui devient l'Unité Temporelle (UT) d'un cycle de base.

La phase de synchronisation, qui dure au maximum 0,48 ms, est nécessaire pour initier le cycle de base : tous les nœuds du système se synchronisent après réception du message d'initialisation. Ce message véhicule, entre autres, le Vecteur d'Adhésion au Groupe (VAG) et le temps qui, selon les caractéristiques du protocole, est soit le temps global, soit le temps local du système. Ce dernier est à initialiser à chaque début de cycle de base.

Après la synchronisation, le nœud intermédiaire appartenant à un niveau de l'architecture, réamplifie les données reçues par le nœud intermédiaire de niveau supérieur, vérifie leur intégrité, les traite éventuellement (suivant le niveau de l'architecture auquel il se trouve) et les retransmet à 1 UT vers un nœud intermédiaire de niveau inférieur, qui émet à son tour des données relatives à l'asservissement à 2 UT.

Cet "aller-retour" s'étend sur 0,96 ms et concerne l'asservissement d'un seul des 10 nœuds intermédiaires connectés au même bus et relatifs à un même niveau.

En faisant les hypothèses que la largeur des fenêtres temporelles de chaque cycle de base est équi-répartie, et que chaque nœud de l'architecture est asservi à chaque cycle de base, il est possible de tirer du tableau certaines conclusions.

Au pire des cas, la latence de transmission des données entre le calculateur et un microspoiler est de 11,04 - 0,48 = 10,56 ms et, au meilleur des cas, elle est de 1,92 - 0,48 = 1,44 ms.

Au final, et vu que l'extrapolation de la contrainte temporelle de "rafraîchissement des surfaces de contrôle toutes les 10 ms" aux nappes de microsurfaces de contrôle est assez "stricte", les principes architecturaux et protocolaires que nous proposons devraient permettre de respecter les contraintes temps réel. D'autant plus que le fait de considérer que les nœuds intermédiaires des niveaux inférieurs sont asservis toutes les 10 ms, et que les microspoilers émettent un retour de position avec cette même périodicité, augmente considérablement le nombre des fenêtres temporelles dans les cycles de base représentant les niveaux inférieurs de l'architecture.

En effet, il est possible de considérer, sans nuire à la robustesse du système de communication, que les nœuds intermédiaires de niveau inférieur ne remontent au niveau supérieur de l'architecture que des messages d'événement. En d'autres termes, ces nœuds peuvent remonter des informations particulières qu'ils n'arrivent pas à gérer soit localement (si l'on n'envisage pas de communication inter-nœuds d'un même niveau), soit avec les nœuds intermédiaires du même niveau. En ce qui concerne les nappes de microsystèmes, et qui peuvent n'avoir que deux états (actif, inactif), on peut considérer qu'ils ne remontent l'information que lorsqu'ils changent d'état ou lorsqu'ils détectent leur défaillance.

Au niveau du cycle de base, ces considérations se traduisent par la suppression de toutes les fenêtres temporelles relatives aux remontées d'informations et leurs remplacements par quelques fenêtres temporelles dédiées à la transmission de messages d'événements. Ainsi, il sera possible d'augmenter le débit utile des données échangées dans le réseau de communication, qui aura une plus grande bande passante. Le rafraîchissement des microsurfaces de contrôle pourrait être ainsi plus fréquent.

De plus, le gain en bande passante pourrait être utilisé pour mettre en œuvre d'autres protocoles ou mécanismes assurant, par exemple, un plus haut niveau d'intégrité des données échangées. Ce gain pourrait également être utilisé pour transmettre des données relatives à des applications hors commandes de vol.

5.4 Problématiques ouvertes

Nos travaux présentés précédemment, concernant le système de communication en vue de la commande de nappes de microactionneurs, doivent être vus davantage comme des pistes que comme une proposition définitive. En effet, il y a encore d'énormes inconnues concernant la technologie des microactionneurs et les lois de commandes associées qui pourront être utilisées dans les futurs avions civils. À notre connaissance, ces deux problématiques sont à ce jour largement ouvertes. Une fois ces inconnues levées, il restera aussi à analyser comment mettre en œuvre la tolérance aux fautes de microactionneurs défaillants au sein d'une nappe de microactionneurs.

5.4.1 Technologie des microactionneurs

Au travers de l'état de l'art que nous avons effectué dans ce chapitre sur l'utilisation de nappes de microsystèmes dans le domaine de l'aéronautique, nous avons vu que le domaine des microactionneurs apparaît plus en retrait que celui des microcapteurs, même s'il existe bien des solutions comme des microflaps. Ces nappes nous semblent pour l'instant davantage destinées à des microdrones qu'à des avions civils.

Nous avons souvent évoqué, dans ce mémoire, le remplacement des surfaces de contrôle actuelles par des nappes de microsurfaces qui pourraient s'inspirer des microflaps. Mais, d'une part, il faut aussi envisager que ces microsurfaces viennent en complément, et non pas en remplacement des surfaces actuelles. Et par ailleurs, il n'est pas tout à fait sûr que l'on fasse au final appel à des microsurfaces pour remplacer ces surfaces actuelles. En effet, la manipulation d'une surface de contrôle actuelle a pour but d'influer sur les forces appliquées à l'avion selon un certain axe. Or, des effets comparables pourraient sans doute également être envisagés avec d'autres types d'actionneurs, comme des nappes de micropropulseurs.

Dans les critères de choix qui permettront de définir le type de microactionneur qui sera retenu, il y a bien sûr ceux du poids, du coût, de la consommation, etc. Mais, il y a surtout ceux de leur fiabilité et de leur robustesse. En effet, il est important de noter que les microactionneurs se trouveront dans un environnement hostile et devront en plus assurer un fonctionnement fiable le plus longtemps possible.

Le type de microactionneur retenu aura aussi un impact sur le moyen d'activation associé (électrique, électromagnétique, etc.). Mais, plus proche de nos préoccupations, ce qu'il est important de connaître, c'est davantage la granularité de la commande associée, que le type d'activation, et une commande en tout ou rien (actif, passif) n'est pas à exclure.

Par ailleurs, le micro-effet associé à un microactionneur devrait permettre de déterminer le nombre de microactionneurs nécessaires pour produire l'effet recherché. Comme il n'est pas à exclure que des microsystèmes incluant un certain nombre de microactionneurs soient développés, c'est surtout le nombre de microsystèmes à gérer qu'il sera important de connaître pour la définition définitive d'un système de communication.

5.4.2 Lois de commande associées

L'utilisation de nappes de microactionneurs en remplacement, et/ou complément, des surfaces de contrôle actuelles aura, sans nul doute, de fortes répercussions sur les lois de commande utilisées pour contrôler le mouvement de l'avion. Tout d'abord, il faudra réfléchir comment tirer profit de la multitude des actionneurs pour améliorer le comportement de l'avion en n'agissant pas forcément de manière uniforme sur l'ensemble de microactionneurs. Il faudra également tenir compte, au niveau de ces lois de commande, du niveau potentiel de granularité de la commande.

Une autre question qui se pose concerne la répartition de la commande entre les différents nœuds intermédiaires du réseau de communication. Les calculateurs de commandes de vol auront sans doute une vue abstraite de l'ensemble des microactionneurs et n'enverront éventuellement des commandes relatives aux effets recherchés qu'au premier niveau de nœuds intermédiaires (cf. figure 5.12). Ce premier niveau de nœuds intermédiaires devrait propager les commandes au niveau inférieur en ajustant éventuellement la commande en fonction de la position du sous-ensemble de microactionneurs concerné. Enfin, dans les derniers niveaux, les nœuds auront peut-être à convertir la commande, relative à l'effet recherché, en un pourcentage de microactionneurs actifs dans le cas de commandes tout ou rien.

Au final, il est évident que l'intelligence de traitement ne se limitera pas à celle des calculateurs, et que c'est bien la question de la distribution de l'intelligence qui se pose.

5.4.3 Tolérance d'un sous-ensemble défaillant de microactionneurs

Nous avons indiqué, précédemment, qu'un des avantages que peut procurer l'utilisation de nappes de microactionneurs est une robustesse plus élevée du système d'actionnement dans la mesure où la perte d'un sous-ensemble de microactionneurs est peu critique. Par exemple, la perte d'un sous-ensemble de microactionneurs correspondant à moins de 1% de l'autorité de contrôle (cf. figure 5.12) ne présente pas un danger vis-à-vis de la commande de vol de l'avion. Par contre, pour conserver sa qualité de vol, il peut être nécessaire de compenser cette perte par une modification des commandes des microactionneurs restants.

L'étude de la tolérance d'un sous-ensemble défaillant de microactionneurs ne peut pas être envisagée tant que des réponses aux problématiques précédentes n'auront pas été apportées. Toutefois, nous pouvons dire que, si l'on veut que la mise en œuvre de cette tolérance soit la plus transparente possible pour les calculateurs de commande, elle nécessitera forcément des communications supplémentaires entre des nœuds intermédiaires d'un même niveau et à l'intérieur d'une même ramification pour répartir différemment la commande.

S'il n'est pas possible de compenser localement la défaillance, des informations devront être retournées vers des nœuds de niveau supérieur, et éventuellement jusqu'aux calculateurs de commande pour arriver à compenser la défaillance à un niveau plus haut.

Enfin, face aux défaillances partielles d'un système d'actionnement, il faudra prévoir des procédures de diagnostic et analyser les répercussions que cela pourra avoir au niveau des stratégies de maintenance de l'avion.

5.5 Conclusion

Dans ce dernier chapitre, nous avons d'abord développé l'état de l'art résumé dans le chapitre 1 sur les microsystèmes, tant individuels que sous forme de grands ensembles. L'objectif de cet état de l'art était d'asseoir nos propos annoncés au début de la présentation de nos travaux. Cet état de l'art a montré l'importante diversité dans les utilisations qui sont faites des microsystèmes, en mettant plus particulièrement en lumière leur très forte émergence dans le domaine avionique. Il a également détaillé les avantages et les problématiques générales.

Dans l'objectif d'intégrer massivement des nappes de microsystèmes dans des applications de commande-contrôle critiques, nous sommes ensuite revenus sur le système de communication qu'il sera nécessaire de mettre en place pour de telles applications, et en ne nous limitant pas cette fois au seul critère que nous avions retenu au cours des chapitres précédents, à savoir le nombre important de microsystèmes, qui pourtant à lui seul justifie la présence de nœuds intermédiaires plus ou moins complexes dans le système de communication.

Nous avons, en particulier, mis en évidence la nécessité de répartir le traitement applicatif et réseau sur les différents niveaux de nœuds intermédiaires. Cela nous a amené à appréhender le système de communication selon trois couches d'abstraction différentes qui se répercutent sur la façon de percevoir la commande : les nœuds intermédiaires des niveaux supérieurs, les nœuds intermédiaires des niveaux inférieurs, les microactionneurs.

Nous avons aussi montré la possibilité d'adapter les moyens assurant l'intégrité des données transmises en fonction du niveau d'autorité de contrôle associé à un niveau de nœuds intermédiaires. Ainsi, la fonction de contrôle d'erreur évolutive proposée dans ce mémoire pourrait n'être utilisée que pour les niveaux supérieurs de l'architecture de communication.

Au cours de ce chapitre, nous avons aussi insisté sur le fait qu'il reste encore beaucoup d'inconnues qui ne permettent pas, à ce stade, de définir une architecture définitive pour la commande de nappes de microsystèmes. Pour lever ces inconnues, il est nécessaire que soient menés des travaux qui impliquent, ou concernent, d'autres domaines comme, par exemple, la technologie des microactionneurs, l'évolution des lois de commande et leur répartition au sein du système de communication.

CONCLUSION GENERALE

Parmi tous les défis que soulève l'utilisation de nappes de microsystèmes dans des systèmes de commande-contrôle critiques, nos travaux avaient pour objectif d'apporter des réponses au défi de la définition d'un système de communication à haut niveau d'intégrité, adapté à la commande de nappes de microactionneurs dans un système de commande-contrôle critique. Ces travaux ont été initiés et guidés par notre cas d'étude, qui est une application du domaine de l'avionique – les systèmes de commandes de vol (CDV) – dans l'optique de pouvoir commander des nappes de plusieurs centaines ou milliers de microsurfaces.

Par rapport à cet objectif, l'étude et la définition de techniques permettant d'obtenir un haut niveau d'intégrité ont en fait constitué une part prépondérante dans nos travaux, d'une part, du fait des insuffisances que nous avons relevées en étudiant certaines de ces techniques existantes, et d'autre part, du fait notamment du nombre important d'inconnues sur les microsystèmes à envisager. Et en définitive, la définition précise d'un système de communication n'est pas encore à maturité.

Néanmoins, pour ce qui est de la solution proposée pour assurer l'intégrité, il a pu être fait abstraction de mises en œuvre spécifiques des couches basses d'un réseau de communication. Ainsi, la solution proposée n'en est que plus générale, tout en répondant aussi, pour notre cas d'étude, à un objectif à court terme, qui concernait le passage à des communications numériques au niveau des CDV.

Ainsi, avant de définir les propriétés que doit garantir une communication intègre, nous avons tout d'abord analysé les caractéristiques des systèmes à commander considérés. Cette analyse a fait ressortir que les CDV étaient représentatifs d'une classe de systèmes que nous avons qualifiés comme étant à dynamique lente et à commande par messages d'état.

Cela nous a, d'une part, conduit à ne considérer que trois des propriétés d'intégrité parmi l'ensemble de celles que nous avons définies d'un point de vue général comme définissant une communication intègre. Nous avons retenu qu'une communication intègre doit garantir qu'un nœud récepteur qui reçoit une commande ne traitera pas une commande : 1) qui est altérée, 2) qui ne lui est pas destiné, 3) qui est périmée. Pour cela, il faut être capable de détecter respectivement les trois types d'erreurs suivants : 1) en valeur, 2) d'adressage, 3) temporelles.

Et d'autre part, le fait de considérer des systèmes à dynamique lente a conduit à considérer que la garantie de ces propriétés sur un message n'a pas besoin d'être très forte, même dans le cas d'un système critique. Car, pour de tels systèmes, l'intégrité est à considérer sur un lot de messages, et ce qui est critique, c'est la répétition au-delà d'un certain seuil de la non détection sur plusieurs messages. Par exemple, c'est la non détection sur 3 messages successifs, ou sur 3 messages dans un lot de 10 messages, qui est susceptible d'engendrer un événement redouté au niveau du système commandé (ex. : l'embarquement de surfaces de contrôle dans les CDV).

Une fois ces hypothèses de travail définies, nous avons analysé le type de système de communication nécessaire à la commande de nappes de microactionneurs. De notre premier tour d'horizon mené sur les contraintes induites par l'utilisation de microactionneurs dans un système de commande-contrôle, nous avons surtout retenu comme facteurs d'impact le nombre élevé des microactionneurs, qui sont invariants en nombre et localisation, et non communicants entre eux. Nous avons également identifié que la commande d'un grand nombre de microactionneurs ne peut pas se satisfaire d'une classique architecture centralisée, mais qu'elle nécessite de faire appel à un système de communication comportant plusieurs types de nœuds intermédiaires, d'une complexité supérieure à celle d'un simple répéteur. La diversité des types de nœuds nous a conduit à considérer comme caractéristiques de ces nœuds qu'ils aient au moins des capacités de mémorisation, si ce n'est de traitement (au moins réseau).

Puis, sur cette architecture réseau, nous avons mené une analyse de risques pour vérifier si le niveau d'intégrité procuré par les fonctions usuelles de contrôle d'erreur mises en œuvre dans de tels équipements (détection d'erreur par code CRC) était compatible avec notre cas d'étude pour lequel le taux d'occurrence de l'événement redouté doit être inférieur à 10^{-9}/h.

Pour cela, nous avons commencé à identifier les différentes sources d'altération des messages et à y associer des modèles d'erreur qui caractérisent à la fois la multiplicité de l'erreur et son étendue temporelle. Les modèles d'erreur associés aux trois principales sources identifiées (bruit, défauts de câblage, défaillance des nœuds intermédiaires) sont respectivement les erreurs : 1) aléatoires indépendantes, 2) en rafale indépendantes et 3) multiples répétitives.

Par la suite, en prenant des taux d'occurrence typiques de chaque source d'altération, et en considérant les codes CRC 16 bits (les plus couramment utilisés), nous avons chiffré le taux d'occurrence de l'événement redouté associé à chaque source d'altération en restreignant notre analyse à la détection des erreurs en valeur. En comparant ces chiffres avec le seuil de criticité visé, nous avons constaté que le niveau d'intégrité recherché n'est pas atteint dans le cas de certains types d'erreurs induites par les défaillances des nœuds intermédiaires. Car, pour ces cas, les erreurs peuvent être répétitives sur plusieurs messages successifs ; et même lorsque les nœuds intermédiaires régénèrent de nouveaux bits de CRC après un traitement réseau, les erreurs survenues lors de ce traitement ne peuvent pas être détectées par le nœud récepteur.

Pour pallier ces insuffisances, nous avons proposé d'utiliser une fonction de contrôle d'erreur de bout en bout, mise en œuvre au niveau de la couche application, et que nous avons appelé "*fonction de contrôle d'erreur évolutive*". L'originalité de notre travail à ce niveau a consisté à proposer une solution permettant d'obtenir un haut niveau d'intégrité, tout en présentant un coût acceptable en nombre de bits de redondance pour un message. Pour cela, nous avons tiré profit du fait que l'intégrité est à considérer sur un lot de messages. Le principe de la solution proposée consiste à changer de manière cyclique (ou autre) de fonction de contrôle pour chaque nouveau message. Il faut donc utiliser plusieurs fonctions de contrôle d'erreur, et les choisir telles qu'elles aient autant que possible des pouvoirs de détection cumulatifs, pour augmenter au mieux le pouvoir de détection de la solution proposée.

Nous avons, par la suite, proposé une mise en œuvre de la fonction de contrôle évolutive à base de codes CRC en montrant tout d'abord que le temps de génération logicielle des bits de CRC est compatible avec les contraintes temporelles de notre cas d'étude. En nous appuyant sur les fondements mathématiques forts sur lesquels reposent les codes CRC, nous avons proposé une méthode pour choisir ces polynômes. Cette méthode se base sur la notion de polynôme primitif et consiste à choisir les polynômes générateurs de façon à ce que le produit des polynômes primitifs qu'ils ont en commun soit de degré le plus petit possible.

Ensuite, pour valider cette mise en œuvre, nous avons développé des modèles Matlab/Simulink. L'ensemble des simulations réalisées nous a permis de montrer que, grâce à la solution proposée, le taux de l'événement redouté pouvait être ramené à une valeur satisfaisant l'objectif visé, c'est-à-dire inférieure à 10^{-9}/h.

Les travaux concernant l'analyse des risques et ceux concernant la définition et la validation d'une fonction de contrôle à haut niveau d'intégrité ont constitué les deux premiers volets de nos travaux.

Le troisième volet a concerné plus spécifiquement les microsystèmes et leurs perspectives d'intégration dans un système de commande-contrôle. Dans ce dernier volet, nous avons d'abord approfondi l'état de l'art des microsystèmes, qu'il s'agisse de microsystèmes individuels, ou sous forme de nappes de microcapteurs ou de microactionneurs. Cet état de l'art a montré l'importante diversité dans les utilisations qui sont faites des microsystèmes, et ce dans de nombreux domaines. Nous avons pu aussi constater que le domaine avionique n'est pas absent de cette mouvance et que de nombreux travaux sont engagés sur ce point.

Nous avons ensuite analysé comment intégrer des nappes de microsystèmes dans un système de commande-contrôle. Nous avons d'abord montré que, compte tenu de la structure arborescente du réseau de commande, il est possible d'associer des niveaux d'autorité différents aux différentes ramifications de l'arbre. Il est alors possible d'adapter les fonctions de contrôle d'erreur utilisées en sein du réseau de commande en fonction du niveau d'autorité des différents nœuds intermédiaires. Ainsi, il suffit d'utiliser la fonction de contrôle évolutive proposée dans ce mémoire, entre le calculateur de commande et les nœuds intermédiaires ayant un niveau d'autorité importante ; des fonctions de contrôle d'erreur avec un pouvoir de détection moins important pouvant être utilisées entre des niveaux ayant une autorité faible. Nous terminons ce volet des travaux en mettant en évidence la nécessité de mener des travaux complémentaires dans d'autres domaines que celui de nos domaines de compétences comme, par exemple, la technologie des microactionneurs, l'évolution des lois de commande, la distribution de la commande.

*

* *

Malgré les travaux présentés dans ce mémoire, l'intégration des nappes de microsystèmes dans des systèmes de commande-contrôle, tels que celui des commandes de vol, reste encore un objectif à long terme. Néanmoins, l'utilisation de la fonction de contrôle à haut niveau d'intégrité proposée dans ce mémoire peut être envisagée à court terme pour protéger les communications numériques au sein de systèmes de commande-contrôle critiques.

Pour cela, il convient toutefois de compléter nos travaux, tout d'abord en affinant le principe de la détection des erreurs d'adressage et des erreurs temporelles, puis en intégrant ce principe dans les modèles de simulation actuels pour pouvoir analyser l'apport de la fonction de contrôle évolutive, vis-à-vis de l'ensemble des erreurs qui sont à considérer pour satisfaire l'ensemble des propriétés d'intégrité à assurer pour la classe de systèmes considérés.

Concernant l'objectif à long terme d'intégration de nappes de microsystèmes dans les systèmes de commandes de vol, un certain nombre de travaux sont à réaliser, dont la plupart concernent ou impliquent d'autres communautés (ou spécialistes d'autres domaines).

Ainsi, il conviendra tout d'abord de travailler sur la technologie même des microactionneurs (pouvoir d'actionnement, consommation, …) en fonction des effets aérodynamiques recherchés. Des compromis seront sans doute à faire entre les effets microscopiques d'un microactionneur de base et le nombre et la position des microactionneurs, nécessaires pour reproduire l'effet macroscopique des surfaces actuelles de contrôle d'un avion, mais aussi, pourquoi pas, pour créer et utiliser d'autres effets.

Des travaux doivent être envisagés au niveau de la commande, en spécifiant de nouvelles lois de commande pour tirer profit au mieux des nappes de microsurfaces de contrôle, et en proposant une répartition de la commande sachant, par exemple, qu'il est fort probable qu'en fin de chaîne de commande, les microsurfaces se commanderont en tout ou rien.

Une autre perspective, ou contrainte, qui en découle est la répartition du traitement applicatif et réseau, par rapport à ces nouvelles lois de commande, ceci entre les différents niveaux de nœuds intermédiaires. Le but étant d'améliorer la commande, ou tout du moins de compenser les défaillances locales sans que le calculateur de commande principal ait à intervenir.

Références bibliographiques

[Adams 2001] C. Adams, “Data Bus Test”, Avionics Magazine, Aviation Today, novembre 2001 (http://www.aviationtoday.com/).

[AEEC 2001] Airlines Electronic Engineering Committee, “Aircraft Data Network, Part 5 - Network Interconnection Devices”, ARINC Specification 664, Draft 1, mai 2001.

[AEEC 2002] Airlines Electronic Engineering Committee, “Aircraft Data Network, Part 2 - Ethernet Physical and Data Link Layer Specification”, ARINC Specification 664, Draft 8, janvier 2002.

[ARINC 2002] “ARINC Protocol Tutorial”, Condor Engineering Inc., mars 2002, 29 p.

[ARINC 2003] “Avionics Application Software Standard Interface - ARINC Specification 653-1”, Aeronautical Radio Inc., Annapolis, Maryland, USA, octobre 2003, 216 p.

[Askerdal *et al.* 2002] O. Askerdal, M. Gäfvert, M. Hiller, N. Suri, “A Control Theory Approach for Analyzing the Effects of Data Errors in Safety-Critical Control Systems”, *Proc. of the 2002 Pacific Rim International Symposium on Dependable Computing (PRDC-2002)*, Tsukuba, Japon, 16-18 décembre 2002, pp. 105-114.

[Baicheva *et al.* 1988] T. Baicheva, S. Dodunekov, P. Kazakov, “On the Cyclic Redundancy-Check Codes with 8-bit Redundancy”, *Computer Communications*, vol. 21, n° 11, août 1998, pp. 1030-1033.

[Baicheva *et al.* 2000] T. Baicheva, S. Dodunekov, P. Kazakov, “Undetected Error Probability Performance of Cyclic Redundancy-Check Codes of 16-bit Redundancy”, *IEEE Proceedings on Communications*, vol. 147, n° 5, octobre 2000, pp. 253-256.

[Bannatyne 1999] R. Bannatyne, “Building Fault Tolerant Embedded Systems using TTP/C”, *Electronics Engineer*.

[Bellare *et al.* 1996a] M. Bellare, R. Canetti, H. Krawczyk, “Message Authentication using Hash Functions -The HMAC Construction”, *RSA Laboratories’ CryptoBytes*, vol. 2, n° 1, Spring 1996.

[Bellare *et al.* 1996b] M. Bellare, R. Canetti, H. Krawczyk, "Keying Hash Functions for Message Authentication", *Advances in Cryptology, - Proc. CRYPTO'96*, Lecture Notes in Computer Science, vol. 1109, Springer Verlag, (Ed. N. Koblitz), 1996, pp. 1-15.

[Bérard 2003] B. Bérard, P. Coupoux, Y. Crouzet, P. David, Y. Garnier, S. Goiffon, G. Mariano, V. Nicomette, L. Planche, I. Puaut, J.-M. Tanneau, H. Waeselynck, "*Logiciel libre et sûreté de fonctionnement – cas des systèmes critiques"*, (Eds P. David et H. Waeselynck), Hermès Science, ISBN 2-7462-0727-3, 2003, pp. 82-88.

[Berlin & Gabriel 1997] A.A. Berlin, K.J. Gabriel, "Distributed MEMS: New Challenges for Computation", *IEEE Computational Sciences & Engineering*, vol. 04, n° 1, janvier-mars 1997, pp. 12-16.

[Bhansali 2001] S. Bhansali, "Probing Human Skin as an Information-rich Biological Interface using MEMS-based Informatics", *Int. Society for Optical Engineering*, 253, p. 14.

[Blanc & Gil 2003] S. Blanc, P.J. Gil, "Improving the Multiple Errors Detection Coverage in Distributed Embedded Systems", *Proc. 2003 International Symposium on Reliable Distributed Systems (SRDS-03)*, Florence, Italie, 6-8 octobre 2003, pp. 303-312.

[Bohringer & Donald 1998] K.F. Bohringer, B.R. Donald, "Micro Contacts and Micro Manipulation with MEMS Actuator Arrays", *IEEE International Conference on Robotics and Automation, Workshop on Modeling Contact Analysis, and Simulation of Mechanical Systems in Robotics and Manufacturing*, Leuven, Belgique, 16-20 mai, 1998.

[Boudreau *et al.* 1994] P.E. Boudreau, W.C. Bergman, D.R. Irvin, "Performance of a Cyclic Redundancy Check and its Interaction with a Data Scramble", *IBM Journal of Research and Development*, vol. 38, n° 6, novembre 1994, pp. 651-658.

[Braun & Waldvogel 2001] F. Braun, M. Waldvogel, "Fast Incremental CRC Updates for IP over ATM Networks", *Proc. 2001 IEEE Workshop on High Performance Switching and Routing (HPSR-2001)*, Dallas, USA, 29-31 mai 2001, pp. 48-52.

[Brière & Traverse 1993] D. Brière, P. Traverse, "AIRBUS A320/A330/A340 Electrical Flight Controls: A Family of Fault-Tolerant Systems", *Proc. 23th International Symposium on Fault-Tolerant Computing (FTCS-23)*, Toulouse, 22-24 juin 1993, pp. 616-623.

[Brodtkorb 2001] H.T. Brodtkorb, "A Safety Layer for Foundation Fieldbus", *Cand. scient. thesis*, University of Oslo, 6 mai 2001.

[Castagnoli *et al.* 1990] G. Castagnoli, J. Ganz, P. Graber, "Optimum Cyclic Redundancy-Check Codes with 16-Bit Redundancy", *IEEE Transactions on Communications*, vol. 38, n° 1, janvier 1990, pp. 111-114.

[Castagnoli *et al.* 1993] G. Castagnoli, S. Bräuer, M. Herrmann, "Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits", *IEEE Transactions on Communications*, vol. 41, n° 6, juin 1993, pp. 883-892.

[Christou 2001] A. Christou, "Materials and Reliability Issues in MEMS and Microsystems", *Proc. 2001 SPIE International Symposium on Microelectronics and Micro-Electro-Mechanical Systems (MICRO/MEMS-2001)*, Adelaïde, Australie, 17-19 décembre 2001, pp. 326-333.

[Coccoli *et al.* 2000] A. Coccoli, S. Schemmer, F. Di Giandomenico, M. Mock, A. Bondavalli, "Analysis of Group Communication Protocols to Assess Quality of Service Properties", *Proc. 5th IEEE International Symposium on High Assurance Systems Engineering (HASE-2000)*, Albuquerque, Nouveau-Mexique, USA, 15-17 novembre 2000, pp. 247-256.

[Collet *et al.* 2004] J.H. Collet, A. de Bonneval, F. Caignet, T. Gayraud, F. Jimenez, R. Plana, C. Rossi, "Réflexion pour une recherche sur les grands ensembles de microsystèmes électromécaniques", *Rapport LAAS n° 04617*, novembre 2004, 80 p.

[Coumar 2003] O. Coumar, "Les systèmes électro-micromécaniques : MEMS et NEMS", *Revue de l'Électricité et de l'Électronique*, septembre 2003, pp. 63-68.

[Cui & Zhou 1996] T.H. Cui, Z.Y. Zhou, "The Application of Micro Field Theory to MEMS Actuators", *Proc. ASME International Mechanical Engineering Congress*, Atlanta, Géorgie, USA, 17-22 novembre 1996, pp. 417-420.

[Curtis & France 1999] H. Curtis, R. France, "Time Triggered Protocol (TTP/C): A Safety-Critical System Protocol", *EE382C Literature Survey*, 24 octobre 1999.

[Curtis 1999] H. Curtis, "Structure and Objectives of MEMS and Microsystems in Europe", *MCC/WTEC Workshop on MicroElectromechanical Systems in Europe*, Maryland, USA, juin 1999.

[Denz & Nilsson 1998] P.R. Dens, A.A. Nilsson, "Performance of Error Control Coding Techniques for Wireless ATM", *Proc. 1998 IEEE International Conference on Communications (ICC-98)*, Atlanta, Géorgie, USA, 7-11 juin 1998, vol. 2, pp. 1099-1103.

[Eccles *et al.* 2001] L.H. Eccles, W. Catlin, M.J. Holland, N.P. Kim, L. Malchodi, "MEMS Pressure Belt with Sensor Interface and Communication Architecture", *Proc. of 8th SPIE International Symposium on Smart Structures and Materials*, Newport Beach, Californie, USA, 5-7 mars 2001, vol. 4334, pp. 309-316.

[Esteve & Campo 2004] D. Esteve, E. Campo "L'électronique des micro/nanosystèmes : évolutions et perspectives", *Revue Signaux*, n° 99, décembre 2004, pp. 9-16

[FAA 2000] Federal Aviation Administration, "System Safety Handbook - chapitre 3 : Principles of System Safety", 30 décembre 2000, 19 p. *(http://www.faa.gov/library/manuals/aviation/risk_management/ss_handbook/media/Chap3_1200.PDF)*

[Fermigier 2004] M. Fermigier, "Couches limites", *Enseignement Hydrodynamique Physique*, Chapitre 8, École Supérieure de Physique et de Chimie Industrielles de Paris, 2004.

[Freeman & Miller 1999] W. Freeman, E. Miller, "An Experimental Analysis of Cryptographic Overhead in Performance-Critical Systems", *Proc. 7th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'99)*, College Park, Maryland, USA, octobre 1999, pp. 348-357.

[Fujwara *et al.* 1985] T. Fujwara, T. Kasami, A. Kitai, S. Lin, "On the Undetected Error Probability for Shortened Hamming Codes", *IEEE Transactions on Communications*, vol. 33, n° 6, juin 1985, pp. 570-574.

[Funk 1996] G. Funk, "Determination of Best Shortened Linear Codes", *IEEE Transactions on Communications*, vol. 44, n° 1, janvier 1996, pp. 1-6.

[Gaujal & Navet 2002] B. Gaujal, N. Navet, "Fault Confinement Mechanisms of the CAN Protocol : Analysis and Improvements", *rapport de recherche INRIA*, RR-4603, octobre 2002, 26 p.

[Geremia 1999] P. Geremia, "Cyclic Redundancy Check Computation: An Implementation Using the TMS320C54x", *Application Report SPRA530*, Digital Signal Processing Solutions, Texas Instruments, avril 1999, 33 p.

[Gilbertson & Busch 1996] R.G. Gilbertson, J.D. Busch, "A Survey of Micro-Actuator Technologies for Future Spacecraft Missions", *Journal of the British Interplanetary Society*, vol. 49, 1996, pp. 129-138.

[Goular *et al.* 2004] D. Goular, D. Fleury, J.-P. Lafforgue, A. Dolfi, J.-P. Cariou, "Caractérisation de tourbillons de sillage par triangulation lidar", *9ème Congrès Francophone de Vélocimétrie Laser*, Université Libre de Bruxelles, 14-17 septembre 2004, pp. H.4.1-H.4.6.

[Grosjean *et al.* 1998] C. Grosjean, G.B. Lee, W. Hong, Y.C. Tai, C.M. Ho, "Micro Balloon Actuators for Aerodynamic Control", *Proc. IEEE Workshop Microelectromechanical Systems*, Heidelberg, Allemagne, 25-29 janvier 1998, pp. 166-171.

[Gue *et al.* 2000] A.M. Gue, D. Estève, V. Conédéra, N. Fabre, "Tête d'injection et de dosage thermique, son procédé de fabrication et système de fonctionnalisation ou d'adressage la comprenant", *Brevet CNRS FR2811588*, Rapport LAAS, n° 000514, juillet 2000, 31 p.

[Hansen 2003] H. Hansen, "Application of Mini-Trailing-Edge Devices in The Awiator Project", *Airbus Deutschland*, EGAG (http://www.kat-net.com/).

[Hartwich *et al.* 2000] F. Hartwich, B. Müller, T. Führer, R. Hugel, "CAN Network with Time Triggered Communication", *Proc. 7th International CAN Conference (ICC)*, Amsterdam, Pays-Bas, 24-25 octobre 2000,7 p. (http://www.can-cia.org/icc/7/index.html)

[Hennebert & Guiho 1993] C. Hennebert, G. Guiho, "SACEM: a Fault-Tolerant System for Train Speed Control", *Proc. 23rd IEEE International Symposium on Fault-Tolerant Computing (FTCS-23)*, Toulouse, 22-24 juin 1993, pp. 624-628.

[Henriksson & Liu 2003] T. Henriksson, D. Liu, "Implementation of Fast CRC Calculation", *Proc. 2003 Asia South Pacific Design Automation Conference*, Kitakyushu, Japon, 21-24 janvier 2003, pp. 563-564.

[Ho *et al.* 1994] C.M. Ho, D. Miu, Y.C. Tai, "Control of Aerodynamic Device by Micro Actuation", *Proc. GOMAC Conference*, San Diego, Californie, USA, novembre 1994, pp. 211-214.

[Ho & Tai 1996] C.M. Ho, Y.C. Tai, "Review: MEMS and its Applications for Flow Control", *Journal of Fluids Engineering*, vol. 118, septembre 1996, pp. 437-447.

[Ho *et al.* 2003] S. Ho, H. Nassef, N. Pornsin-Sirirak, Y.-C. Tai, C.-M. Ho "Unsteady Aerodynamics and Flow Control for Flapping wing Flyers", *Progress in Aerospace Sciences*, vol. 39, n° 8, novembre 2003, pp. 635–681.

[Huang 2001] A. Huang *et al.*, "Gryphon M3 system: Integration of MEMS for Flight Control", *Proc. SPIE (Int'l Society for Optical Engineering)*, vol. 4559, San Francisco, Californie, USA, octobre, 2001, pp. 85-94.

[Huang *et al.* 2001] A. Huang, C. Folk, C. Silva, B. Christensen, Y.F. Chen, G.B. Lee, M. Chen, S. Newbern, F. Jiang, C. Grosjean, C.M. Ho, and Y.C. Tai, "Applications of MEMS Devices to Delta Wing Aircraft: From Concept Development to Transonic Flight Test", *39th AIAA Aerospace Sciences Meeting,* Reno, Nevada, USA, 8-11 janvier 2001.

[Judy & Muller 1997] J.W. Judy, R.S. Muller, "Magnetically Actuated, Addressable Microstructures", *Journal of Microelectromechanical Systems,* vol. 6, n° 3, septembre 1997, pp. 249-256.

[Judy & Motta 2002] J.W. Judy, P.S. Motta, "Introduction to Micromachining and MEMS", *International Conference on Engineering Education,* Manchester, Angleterre, 18-21 août 2002.

[Jumel *et al.* 2003] F. Jumel, N. Navet, F. Simonot-Lion, "Influence des performances d'une architecture informatique sur la fiabilité des systèmes échantillonnés", *Proc. 11th International Conference on Real-Time and Embedded Systems (RTS'03),* Paris, 1-3 avril 2003.

[Kandasamy *et al.* 2002] N. Kandasamy, J.P. Hayes, B.T. Murray, "Time-Constrained Failure Diagnosis in Distributed Embedded Systems", *Proc. 2002 International Conference on Dependable Systems and Networks (DSN-2002),* Washington, DC, USA, 23-26 juin 2002, pp. 449- 458.

[Kasami *et al.* 1985] T. Kasami, T. Fujiwara, H. Shu-Lin, "An Approximation to the Weight Distribution of Binary Linear Codes", *IEEE Transactions on Information Theory*, vol. 31, n° 6, novembre 1985, pp. 769-779.

[Katnet 1997] "AEROMEMS – An Investigation into the Viability of MEMS Technology for Boundary Layer Control on Aircraft", *Project Brite-EuRam III*, n° BE97-4294, (http://www.kat-net.com/projects/display.php?id=6).

[Katnet 2002] "AEROMEMS II – Advanced Aerodynamic Flow Control Using MEMS", (http://www.kat-net.com/projects/display.php?id=5).

[Kazakov 2001] P. Kazakov, "Fast Calculation of the Number of Minimum-Weight Words of CRC Codes", *IEEE Transactions on Information Theory,* vol. 47, n° 3, mars 2001, pp. 1190-1195.

[Kim *et al.* 2001] N.P. Kim, M.J. Holland, C.-P. Chien, M.H. Tanielian, J. Wu, C.P. Wong, "Aircraft Flight Tests and Reliability Improvements of MEMS Pressure Sensor Assembly", *Journal of Surface Mount Technology,* vol. 14-1, janvier 2001, pp. 1-8.

[Kimura *et al.* 1997] M. Kimura, S. Tung, C.-M. Ho, F. Jiang, Y.C. Tai, "MEMS for Aerodynamic Control", *Proc. 28th AIAA Fluid Dynamics Conference,* Snowmass Village, Colorado, USA, 29 juin - 2 juillet 1997, paper 97-2118 (9 p).

[Koopman 2002] P. Koopman, "32-Bit Cyclic Redundancy Codes for Internet Applications", *Proc. International Conference on Dependable Systems and Networks (DSN-2002),* Washington DC, USA, 23-26 juin 2002, pp. 459- 468.

[Kopetz & Ochsenreitezr 1987] H. Kopetz, W. Ochsenreitezr, "Clock Synchronization in Distributed Real-Time Systems", IEEE Transactions on Computers, vol. 36, n° 8, août 1987, pp. 933-940.

[Kopetz 1997] H. Kopetz, "Real-Time Systems, Design Principles for Distributed Embedded Applications", *Kluwer Academic Publishers,* Boston, Massachusetts, avril 1997, 356 p.

[Kopetz 1998] H. Kopetz, "A Comparison of CAN and TTP", *Proc. 15th IFAC Workshop on Distributed Computer Control Systems (DCCS-98),* Como, Italie, 9-11 septembre 1998, pp. 117-128.

[Kopetz & Bauer 2003] H. Kopetz, G. Bauer, "The Time-Triggered Architecture", *Proceedings of the IEEE,* Special Issue on Modeling and Design of Embedded Software, janvier 2003, pp. 112-126.

[Krishnamoorthy *et al.* 2001] U. Krishnamoorthy, K. Li, K. Yu, D. Lee, J.P. Heritage, O. Solgaard, "Self-Aligned Vertical Comb-Drive Actuators for Optical Scanning Micromirrors", *Proc. 2001 IEEE/LEOS International Conference on Optical MEMS,* Okinawa, Japon, 25-28 septembre 2001, pp. 41-42.

[Kumar *et al.* 1998] S.M. Kumar, W.C. Reynolds, T.W. Kenny, "MEMS Based Actuators for Boundary Layer Control", *Proc. 1998 ASME Winter Meeting,* Anaheim, Californie, USA, vol. 66, 15-20 novembre 1998, pp. 103-109.

[Laprie *et al.* 1996] J.C. Laprie, J. Arlat, J.P. Blanquart, A. Costes, Y. Crouzet, Y. Deswarte, J.C. Fabre, H. Guillermain, M. Kaaniche, K. Kanoun, C. Mazet, D. Powell, C. Rabejac, P. Thevenod-Fosse, "Guide de la sûreté de fonctionnement", *Cépadues Éditions,* N°ISBN 2-85428-382-1, 1995, 324 p.

[Laprie 2004] J.-C. Laprie, "Sûreté de fonctionnement des systèmes : concepts de base et terminologie", *Revue de l'Électricité et de l'Électronique,* n° 11, décembre 2004, pp. 95-105.

[Liu & Layland 1973] C. L. Liu, J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environnement", *Journal of the ACM,* vol. 20, n° 1, janvier 1973, pp. 46-61.

[Liu *et al.* 1995] C. Liu, T. Tsao, Y.C. Tai, W. Liu, P. Will, C.M. Ho, "A Micromachined Permalloy Magnetic Actuator Array for Micro Robotics Assembly Systems", *Proc. 8th International Conference on Solid-State Sensors and Actuators (Transducers 95),* Stockholm, Suède, vol. 1, 25-29 juin 1995, pp. 328-331.

[Lyshevski 1999] S.E. Lyshevski, "Electromechanical Flight Actuators for Advanced Flight Vehicles", *IEEE Transactions on Aerospace and Electronic Systems,* vol. 35, n° 2, avril 1999, pp. 511-518.

[Lyshevski 2001] S.E. Lyshevski, "Distributed Control of MEMS-Based Smart Flight Surfaces", *Proc. 2001 American Control Conference,* Arlington, VA, USA, 25-27 juin 2001, pp. 2351-2356.

[Lyshevski 2002] S.E. Lyshevski, "Smart Flight Control Surfaces with Microelectromechanical Systems", *IEEE Transactions on Aerospace and Electronic Systems,* vol. 38, n° 2, avril 2002, pp. 543-552

[Merley & Posner 1984] P. Merley, E.C. Posner, "Optimum Cyclic Redundancy Codes for Noisy Channels", *IEEE Transactions on Information Theory,* vol. 30, n° 6, novembre 1984, pp. 865-867.

[Microchip 2001] Microchip, "PIC18CXX2 Data Sheet High Performance Microcontrollers with 10-bit A/D", *Microchip Technology Inc.*, DS39026C, 2001, 305 p.

[Miller *et al.* 1998] S.L. Miller, M.S. Rodgers, G. LaVigne, J.J. Sniegowski, P. Clews, D.M. Tanner, K.A. Peterson, "Failure Modes in Surface Micromachined MicroElectro-Mechanical Actuators", *Proc. of 36th IEEE International Reliability Physics Symposium (IRPS-1998)*, Reno, Nevada, USA, 31 mars - 2 avril 1998, pp. 17-25.

[Nathanson *et al.* 1967] H.C. Nathanson, W.E. Newell, R.A. Wickstrom, J.R. Davis, "The Resonant Gate Transistor", *IEEE Transactions on Electronic Devices*, vol. 14, n° 3, mars 1967, pp. 117-133.

[Navet *et al.* 2005] N. Navet, Y. Song, F. Simonot-Lion, C. Wilwert, "Trends in Automotive Communications Systems", *Proceedings of the IEEE*, Special Issue on Industrial Communications Systems, vol. 93, n° 6, juin 2005, pp. 1204-1223.

[Paulitsch *et al.* 2005] M. Paulitsch, J. Morris, B. Hall, K. Driscoll, E. Latronico, P. Koopman, "Coverage and the Use of Cyclic Redundancy Codes in Ultra-Dependable Systems", *Proc. 2005 International Conference on Dependable Systems and Networks (DSN-2005)*, Yokohama, Japon, 28 juin - 1 juillet 2005, pp. 346-355.

[Peterson 1961] W.W. Peterson, "Error Correcting Codes", *MIT Press*, 1961

[Peterson & Weldon 1972] W.W. Peterson, E.J. Weldon, "Error-Correcting Codes", *MIT Press*, deuxième édition, Cambridge, MA, USA, 1972.

[Pfeifer *et al.* 1999] H. Pfeifer, D. Schwier, F.W. von Henke, "Formal Verification for Time-Triggered Clock Synchronization", *Proc. 7th IFIP Working Conference on Dependable Computing for Critical Applications (DCCA-7)*, San Jose, Californie, USA, 6-8 janvier 1999, pp. 207-226.

[Poledna *et al.* 2001] S. Poledna, W. Ettlmayr, M. Novak, "Communication Bus for Automotive Applications", *Proc. 27th European Solid-State Circuits Conference (ESSCIRC-2001)*, Villach, Autriche, 18-20 septembre 2001, pp. 482-485

[Powell 2001] D. Powell, "A Generic Fault-Tolerant Architecture for Real-Time Dependable Systems", *Kluwer Academic Publishers*, Ed. D. Powell, 242 p.

[Ramabadran & Gaitonde 1988] T.V. Ramabadran, S.S. Gaitonde, "A Tutorial on CRC Computations", *IEEE Micro*, vol. 8, n° 4, août 1988, pp. 62-75.

[Randell & Muller 2000] C. Randell, H. Muller, "Context Awareness by Analyzing Accelerometer Data", *Proc. 4th International Symposium on Wearable Computers*, Atlanta, Georgia, USA, 16-17 octobre 2000, pp. 175-176.

[RFC 791] Request For Comments (RFC) 791, "Internet Protocol", *Information Sciences Institute*, J. Postel (éditeur), Univ. of Southern California, septembre 1981, http://www.ietf.org/rfc/rfc0791 (version française disponible sur http://abcdrfc.free.fr/rfc-vf/pdf/rfc791.pdf).

[RFC 3385] D. Sheinwald, J. Staran, P. Thaler, V. Cavanna, "Internet Protocol Small Computer System Interface (iSCSI) Cyclic Redundancy Check (CRC)/Checksum Considerations", *RFC 3385*, septembre 2002 (http://www.ietf.org/rfc/rfc3385).

[Rolin 1990] P. Rolin, "Réseaux locaux, normes et protocoles", *Éditions Hermès*, ISBN 2-86601-234-8.

[Rossi *et al.* 1999] C. Rossi, D. Estève, N. Fabre, T. Do Conto, V. Conédéra, D. Dilhan, Y. Guélou, "A New Generation of MEMS based Microthrusters for Microspacecraft Applications", *Proc. International Conference on Micro/Nanotechnology for Space Applications (MNT-1999)*, Pasadena, Californie, USA, 10-15 avril 1999, pp. 201-209.

[Rushby 2003] J. Rushby, "A Comparison of Bus Architectures for Safety-Critical Embedded Systems", *Technical Report Computer Science Laboratory n° NASA/CR-2003-212161*, SRI International, Menlo Park, CA, USA, mars 2003, 63 p.

[SAE Handbook 1995] "Class C Application Requirements, Survey of Known Protocols J20056", *SAE Handbook*, SAE Press, Warrendale, PA, USA, pp. 23.437-23.461

[Saxena & McClukey 1989] N. Saxena, E.J. McClukey, "Arithmetic and Galois Checksums", *Proc. IEEE International Conference on Computer-Aided Design (ICCAD-1989)*, Santa Clara, CA, USA, 5-9 novembre1989, pp. 570-573.

[Schmidt 2000] T. Schmidt, "CRC Generating and Checking", *Microchip Technology Inc.*, Application Note 730, 22 mai 2000, 22 p.

[Sivencrona *et al.* 2000] H. Sivencrona, J. Hedberg, O. Bridal, "Design Principles for Dependable Time-Triggered Control Systems", *PALBUS Project*, report n° 10.7, Department of Computer Engineering, Chalmers, Suède, 13 décembre 2000, 32 p.

[Smith 1981] T.B. Smith, "Fault-Tolerant Clocking System", *Proc. 11th International Symposium on Fault-Tolerant Computing (FTCS-11)*, Portland, Maine, USA, juin 1981, pp 262-264.

[Spitzer 2000] C.R Spitzer, "Avionics Handbook", *CRC Press*, Catalog Number 8348, ISBN:084938348X, décembre 2000, 576 p.

[Stark 2002] P. Stark, "Cyclic Redundancy Check (CRC)", *Educational Materials*, Communications 101 Textbook, Appendix D, 7 p. (http://www.users.cloud9.net/~stark/commbook.htm).

[Stone *et al.* 1998] J. Stone, M. Greenwald, C. Partridge, J. Hughes, "Performance of Checksums and CRCs over Real Data", *IEEE/ACM Transactions on Networking*, vol. 6, n° 5, octobre 1998, pp. 529-543.

[Tabeling 2001] P. Tabeling, "Les Micro-Systèmes", Bulletin de la SFP, juin 2001, 4 p.

[Tanner 2000] D.M. Tanner, "Reliability of Surface Micromachined MicroElectro-Mechanical Actuators", Proc. of 22nd International Conference on Microelectronics (MIEL-2000), Nis, Serbie, 14-17 mai 2000, pp. 97-104.

[Thielicke & Obermeier 2000] E. Thielicke, E. Obermeier, "Microactuators and their Technologies", *Mechatronics*, vol. 10, n° 4-5, juin 2000, pp. 431-455.

[Tindell *et al.* 1995] K. Tindell, A. Burns, J. Wellings, "Analysis of Hard Real-Time Communications", Real-Time Systems, vol. 9, n° 2, septembre 1995, pp. 147-171.

[Totel *et al.* 1998] E. Totel, J.-P. Blanquart, Y. Deswarte, D. Powell, "Supporting Multiple Levels of Criticality", *Proc. 28th Annual International Symposium on Fault-Tolerant Computing (FTCS-28)*, Munich, Allemagne, 25-27 juin 1998, pp. 70-79.

[Traverse *et al.* 2004] P. Traverse, I. Lacaze, J. Souyris, "Airbus Fly-by-Wire: A Total Approach to Dependability", *Proc. 18th IFIP World Computer Congress*, Toulouse, 22-27 août 2004, pp. 191-212.

[TTTECH 1999] "Specification of the TTP/C Protocol", *TTTech website*. (http://www.tttech.com)

[Vinter *et al.* 2001] J. Vinter, J. Aidemark, P. Folkesson, J. Karlsson, "Reducing Critical Failures for Control Algorithms Using Executable Assertions and Best Effort Recovery", *Proc. 2001 International Conference on Dependable Systems and Networks (DSN-2001)*, Göteborg, Sweden, 1-4 juillet 2001, pp. 347-356.

[Weinberg 1999] H. Weinberg, "Dual Axis, Low g, Fully Integrated Accelerometers", *Analog Dialogue*, vol. 33, n° 1, janvier 1999, pp. 23-24.

[Wilwert *et al*. 2003] C. Wilwert, Y. Song, F. Simonot-Lion, T. Clément, "Evaluating Quality of Service and Behavioral Reliability of Steer-by-Wire Systems", *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA'03)*, Lisbonne, Portugal, 16-19 Septembre 2003, pp. 193-200.

[Wilwert 2003] C. Wilwert, "Influence des fautes transitoires et des performances temps réel sur la sûreté des systèmes X-by-Wire", *Doctorat de l'Institut National Polytechnique de Lorraine*, 24 mars 2005.

[Witzke & Leung 1985] K.A. Witzke, C. Leung, "A Comparison of Some Error Detecting CRC Code Standards", *IEEE Transactions on Communications*, vol. 33, n° 9, septembre 1985, pp. 996-998.

[Yang & Han 2002] Y.C. Yang, K.S. Han, "Damage Monitoring and Impact Detection using Optical Fibber Vibration Sensors", *Smart Materials and Structures*, vol. 11, n° 3, juin 2002, pp. 337–345.

[Youssef *et al.* 2003a] A. Youssef, A. de Bonneval, Y. Crouzet, J.J. Aubert, P. Brot, Premier rapport d'avancement du projet "Commande de Vol du Futur (CVF)", *Rapport LAAS n° 03540*, février 2003, 72 p.

[Youssef *et al.* 2003b] A. Youssef, A. de Bonneval, Y. Crouzet, "Définition et évaluation d'une nouvelle architecture de communication pour les systèmes de commande de vol", *Manifestation des Jeunes Chercheurs dans le domaine des Sciences et Technologies de l'Information et de la Communication (MAJESTIC-2003)*, Marseille, 29-31 octobre 2003, 6 p.

[Youssef *et al.* 2004] A. Youssef, A. de Bonneval, Y. Crouzet, J.J. Aubert, P. Brot, "Détection d'erreurs dans les données concernant l'actionnement d'un organe de véhicule", *demande de brevet d'invention*, n° d'enregistrement national 04 12141.

[Youssef *et al.* 2005] A. Youssef, A. de Bonneval, Y. Crouzet, "Définition et évaluation d'un système de communication pour des systèmes de commande intégrant des nappes de MEMS", *3ème Conférence internationale des Sciences Électroniques, Technologies de l'Information et des Télécommunications (SETIT-2005)*, Sousse, Tunisie, 27-31 mars 2005, Actes sur CD-ROM, N° ISBN : 9973-51-546-3.

Annexe A - Mise en œuvre logicielle des codes CRC

Cette annexe présente le fonctionnement, et le code assembleur, des trois algorithmes standard de **mise en œuvre logicielle de codes CRC** cités dans le chapitre 4 :

1) Cyclic Redundancy Code Bitwise (CRCB),
2) Cyclic Redundancy Code Table lookup (CRCT),
3) Cyclic Redundancy Code Reduced table lookup (CRCR).

A.1 Cyclic Redundancy Code Bitwise (CRCB)

Il s'agit d'une mise en œuvre logicielle du circuit matériel à base de registres à décalage.

A.1.1 Notations

- R : contenu du registre CRC à *r* bits et le MSB est noté R_{r-1}
- G : les *r* coefficients du polynôme générateur et G_{r1} = G décalé à droite d'un bit
- i : le bit en entrée pour chaque cycle de calcul
- ^ : Opération *Ou Exclusive* (XOR)
- << n : Décalage à gauche de *n* bits

A.1.2 Algorithme

1- Initialisation de R à 0

2- Boucler pour tout bit en entrée *i*

2-1 Si (R_{r-1} == 1) R = ((R ^ G_{r1}) << 1) + (i ^1)

2-2 Sinon R = (R << 1) + i

A.1.3 Code assembleur

CRC-CCITT (1+ x5 + x12 + x16) Computation
An implementation using the TMS320C54x 14 (Texas Instruments Incorporated)

Partie commune à tous les codes

```
Stack setup and Reset vector
BOS     .usect  "stack",0fh             ; setup stack
TOS     .usect  "stack",1               ; Top of stack at reset
        .sect   "vectors"
Reset:
        bd      Entry                   ; reset vector
        stm     #TOS,SP                 ; Setup stack
```

Main Program

```
        .sect   "program"
Entry
        .include "c5xx.inc"
        ld      #crc,DP                 ; scratch pad mem -> DP=0
        portr   #inport,@nbDataIn       ; nb words
        addm    #-1,@nbDataIn           ; for repeat
```

CRCB (word wise) : bit-wise CRC calculation

```
        stm     #input,AR2              ; copy input words in RAM
        rpt     @nbDataIn               ;
        portr   #I      nport,*AR2+     ; read them from IO space
        ld      #genCRC,16,B            ; BH = CRC generator polynomial
        mvdm    @nbDataIn,AR1           ; AR1 = length of message
        stm     #input,AR2              ; AR2 points to input word
        ld      *AR2+,16,A              ; initialize CRC register
next    calld   CRCB                    ; perform CRC computation
        or      *AR2+,A
        nop
        banz    next,*AR1-              ; process all input words
        sth     A,@crc                  ; store result in memory
```

CRCB routine

Input
1) 16-bit input words in AL
2) CRC generator polynomial in BH
3) OVM = 0

Ouput
CRC value in AH
Code size = 10 words
Cycles = 11 + 5*N = 91 for N = 16 bits

A = [A31….....A16 A15.....….....A0] B = [B31.....B16 B15….....B0]
<------ CRC -----> <---- input bits-----> <--polynomial-->

```
CRCB    stm     #16-1,BRC               ; BRC = 16-1
        rptb    CRC_end-1               ; repeat block
        sftl A,1,A                      ; A = A << 1, C=MSB
        PIPELINE
        PIPELINE
        xc      1,C ;
        xor     B,A                     ; if C=1,
                                        ; AH = new CRC
                                        ; = AH XOR gen.
CRC_end ret                             ; end of repeat
```

A.2 Cyclic Redundancy Code Table lookup (CRCT)

CRCT se base sur un tableau contenant des valeurs pré-calculées représentant les restes des divisions polynomiales d'une série exhaustive de polynômes de même degré par le polynôme générateur du code CRC ; par exemple, pour un CRC 16 bits, ce tableau contient 256 valeurs.

A.2.1 Notations et représentations

- S'(x) : le contenu du registre CRC (s_0 …. s_{15})

$$S'(x) = \underbrace{R_{g(x)}(t_0x^{16} + t_1x^{17} + \ldots\ldots + t_7x^{23})}_{X_1} + \underbrace{(s_0x^8 + s_1x^9 + \ldots\ldots + s_7x^{15})}_{X_2}$$

- $R_{g(x)}$ (A(x)) : le reste de la division polynomiale de *A(x)* par *g(x)*
- $t_i = b_i + s_i + 8$ (pour $i \in [0, 7]$)
- b_i (pour $i \in [0, 7]$) : les données en entrée
- ^ : Opération *Ou Exclusif* (XOR)

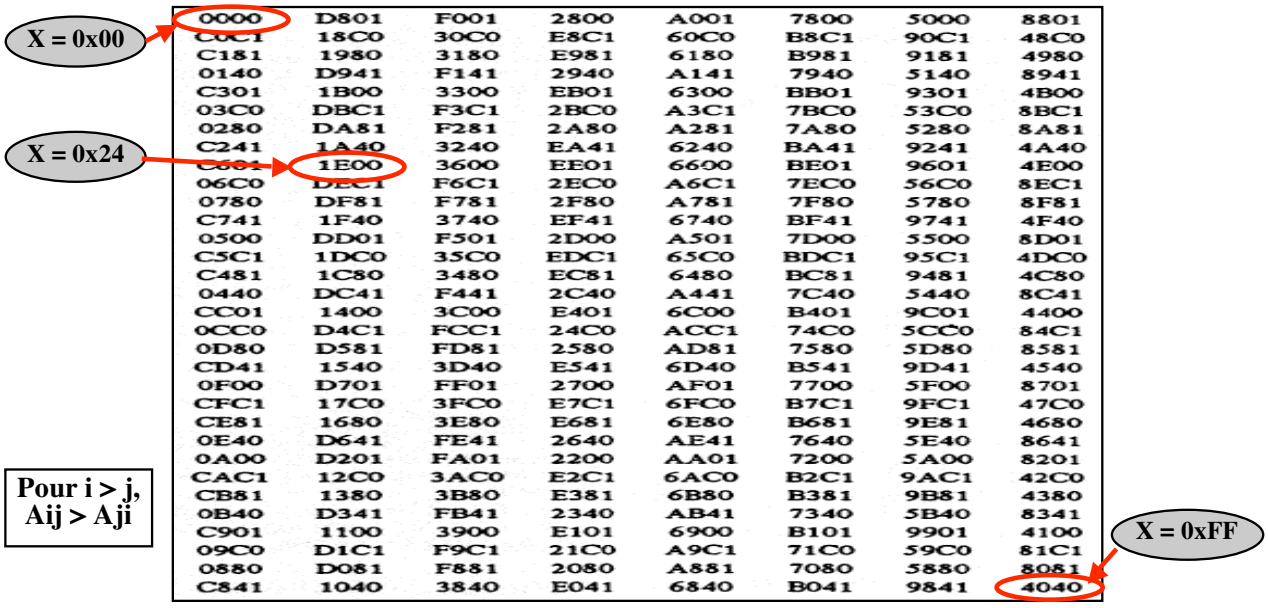
A.2.2 Algorithme

1- Initialisation du registre CRC à 0000_{hexa} ➜ $(s_0 \ldots. s_{15}) = (0 \ldots.0)$
2- Boucler pour tout octet en entrée $(b_0, b_1, \ldots., b_7)$
  2-1$(b_0, b_1, \ldots., b_7)$ ^ $(s_8, s_9, \ldots. , s_{15})$ ➜ $(t_0, t_1,\ldots., t_7)$
  2-2 Décaler le registre CRC de 8 positions à droite ➜ X_2
  2-3 Dégager la valeur relative à $(t_0, t_1,\ldots., t_7)$ du tableau : Lookup Table (contenant des valeurs pré-calculées : restes de division polynomiale) ➜ X_1
  2-4 La valeur dégagée ^ registre CRC ➜ S'(x)

A.2.3 Exemple de Lookup Table (pour un CRC 16 bits)

- Les valeurs de ce tableau (lookup table) correspondent à $R_{g(x)}(X)$ pour $X \in [0x00, 0xFF]$
- Il y a donc 256 restes de division polynomiale : $R_{g(x)}$ en notation hexadécimale classés par ordre croissant selon les colonnes puis les lignes

A.2.4 Code assembleur



0000	D801	F001	2800	A001	7800	5000	8801
C0C1	18C0	30C0	E8C1	60C0	B8C1	90C1	48C0
C181	1980	3180	E981	6180	B981	9181	4980
0140	D941	F141	2940	A141	7940	5140	8941
C301	1B00	3300	EB01	6300	BB01	9301	4B00
03C0	DBC1	F3C1	2BC0	A3C1	7BC0	53C0	8BC1
0280	DA81	F281	2A80	A281	7A80	5280	8A81
C241	1A40	3240	EA41	6240	BA41	9241	4A40
C001	1E00	3600	EE01	6600	BE01	9601	4E00
06C0	DEC1	F6C1	2EC0	A6C1	7EC0	56C0	8EC1
0780	DF81	F781	2F80	A781	7F80	5780	8F81
C741	1F40	3740	EF41	6740	BF41	9741	4F40
0500	DD01	F501	2D00	A501	7D00	5500	8D01
C5C1	1DC0	35C0	EDC1	65C0	BDC1	95C1	4DC0
C481	1C80	3480	EC81	6480	BC81	9481	4C80
0440	DC41	F441	2C40	A441	7C40	5440	8C41
CC01	1400	3C00	E401	6C00	B401	9C01	4400
0CC0	D4C1	FCC1	24C0	ACC1	74C0	5CC0	84C1
0D80	D581	FD81	2580	AD81	7580	5D80	8581
CD41	1540	3D40	E541	6D40	B541	9D41	4540
0F00	D701	FF01	2700	AF01	7700	5F00	8701
CFC1	17C0	3FC0	E7C1	6FC0	B7C1	9FC1	47C0
CE81	1680	3E80	E681	6E80	B681	9E81	4680
0E40	D641	FE41	2640	AE41	7640	5E40	8641
0A00	D201	FA01	2200	AA01	7200	5A00	8201
CAC1	12C0	3AC0	E2C1	6AC0	B2C1	9AC1	42C0
CB81	1380	3B80	E381	6B80	B381	9B81	4380
0B40	D341	FB41	2340	AB41	7340	5B40	8341
C901	1100	3900	E101	6900	B101	9901	4100
09C0	D1C1	F9C1	21C0	A9C1	71C0	59C0	81C1
0880	D081	F881	2080	A881	7080	5880	8081
C841	1040	3840	E041	6840	B041	9841	4040

CRCT (Byte wise) : standard lookup table CRC calculation

```
        stm     #input,AR2          ; AR2 points to input word
        mvdm    @nbDataIn,AR1       ; AR1 = length of message
        ld      #0,A                ; clear Acc A
        rsbx    SXM                 ; no sign extension
        stm     #t_start,AR3        ; AR3 = LTU start address
next1   calld   CRCT                ; process LSByte
        ld      *AR2,-8,B           ; BL = LSByte
        ld      *AR2+,B
        calld   CRCT                ; process MSByte
        and     #0FFh,B             ; BL = MSByte
        banz next1,*AR1-
        stl     A,@crc              ; store result
```

CRCT routine : standard lookup table CRC calculation with bytes as input

Input
1) 8-bit words in AL
2) AR3 = LTU start address
3) DP = 0 = DP(temp)
4) SXM = 0

Ouput
CRC value in AL
Code size = 10 words
Cycles = 12 cycles

A = [A31….....….A16 A15.....….....A0] B = [B31.....…….B7….....B0]
<------ CRC -------> <---input--->

```
CRCT
        stl     A,-8,@AR4       ; AL (low part of Acc A) = CRC bits
                                ; AR4 = CRC >> 8
        stl     A,8,@temp       ; temp = CRC << 8
        xor     @AR4,B          ; B = (input Byte) XOR (CRC>>8)
                                ;     = Offset in LTU
        adds    @AR3,B          ; B = Offset + LTU start address
        stlm    B,AR4           ; AR4 = absolute address in LTU
        retd
        ld      @temp,A         ; AL = CRC << 8
        xor     *AR4,A          ; AL = new CRC
CRCT_end ret
```

A.3 Cyclic Redundancy Code Reduced table lookup (CRCR)

CRCR se base sur un tableau réduit de valeurs pré-calculées représentant les restes de divisions polynomiales d'une série de polynômes particuliers sous la forme x^i par le polynôme générateur. Par exemple, pour un CRC 16 bits, ce tableau contient 8 valeurs.

A.3.1 Notations et représentations

- S'(x) : le contenu du registre CRC (s_0 …. s_{15})
- $R_{g(x)}$ (A(x)) : le reste de la division polynomiale de A(x) par g(x)
- $t_i = b_i + s_i + 8$ (pour $i \in [0, 7]$)
- b_i (pour $i \in [0, 7]$) : les données en entrée
- ^ : Opération *Ou Exclusive* (XOR)

$$S'(x) = \underbrace{R_{g(x)}\,(\,t_0x^{16} + t_1x^{17} + \ldots\ldots\ldots + t_7x^{23})}_{X_1} + \underbrace{(s_0x^8 + s_1x^9 + \ldots\ldots\ldots\ldots + s_7x^{15})}_{X_2}$$

$$\underbrace{R_{g(x)}\,(\,t_0x^{16})}_{X_{1,1}} + R_{g(x)}\,(t_1x^{17}) + \ldots\ldots\ldots + \underbrace{R_{g(x)}\,(t_7x^{23})}_{X_{1,8}}$$

A.3.2 Algorithme

1- Initialisation du registre CRC à 0000_{hexa} ➜ (s_0 …. s_{15}) = (0 ….0)
2- Boucler pour tout octet en entrée (b_0, b_1 , …., b_7)

2-1 (b_0, b_1 , …., b_7) ^ (s_8, s_9, …. , s_{15}) ➜ (t_0, t_1,….., t_7)
2-2 Décaler le registre CRC de 8 positions à droite ➜ X_2
2-3 Boucler pour i = 0, 1, …., 7
2-3-1 SI t_i = 1 dégager la valeur correspondante à l'indice i du tableau : Reduced *lookup Table* (contenant les restes de division polynomiale $R_{g(x)}(x^{i+16})$) ➜ $X_{1,i}$
2-3-2 La valeur dégagée ^ registre CRC ➜ S'(x)

A.3.3 Exemple de Reduced Table Lookup Table (pour un CRC 16 bits)

- Les valeurs de cette table correspondent à $R_{g(x)}(x^{16+i})$ pour $i \in [0, 7]$.
- Il y a donc 8 restes de division polynomiale pour chaque valeur de l'indice i, en notation hexadécimale.

Degré de x^i	**$R_{g(x)}(x^i)$ (en hexa)**
16	A001
17	F001
18	D801
19	CC01
20	C601
21	C301
22	C181
23	C0C1

A.3.4 Code assembleur

CRCR (Word wise) : Reduced lookup table CRC calculation

```
        stm     #input,AR2              ; AR2 points to input word
        mvdm    @nbDataIn,AR1           ; AR1 = length of message
        ld      #0,A                    ; clear Acc A
        stm     #rtw_start-1,AR3        ; AR3=LTU start address-1
next2   xor     *AR2+,A                 ; AL = CRC XOR input word
        calld   CRCR                    ; process input word
        and     #0FFFFh,A
        banzd    next2,*AR1-
        stm     #rtw_start-1,AR3        ; AR3=LTU start address-1

        stl     A,@crc                  ; store result
done    b       done
```

CRCR routine : Reduced lookup table CRC calculation with words as input

Input
1) 16-bit words in AL
2) AR3 = LTU start address -1

Ouput
CRC value in AL
Code size = 15 words
Cycles = 11 + 5*N = 91 for N = 16 bits
A = [A31…....…….A16 A15….....……A0]
<---------- CRC ---------->

CRCR

```
        stm     #16-1,BRC       ; BRC = 16-1
        rptbd   #loop-1         ; repeat block
        ld      #0,B            ; reset B
        nop
          ror   A               ; get MSBit in Carry
          mar   *AR3+           ; increment index in LTU
          nop
          xc    1,C             ; test Carry
          xor   *AR3,B          ; if 1 then B = B XOR table[i++]
loop    retd                    ; end of repeat
        xor     B,A             ; AL = new CRC
        nop

        .bss    input,257       ; augmented message (16 bits appended)
crc     .usect  "scratchPad",32 ; scratch pad memory
temp    .set    crc+1
nbDataIn .set   temp+1
```

CRCR_end ret

```
        .sect   "reducedTablew" ; reduced LUT
                                ; (based on words)
rtw_start
        .word   01021h
        .word   02042h
        .word   04084h
        .word   08108h
        .word   01231h
        .word   02462h
        .word   048C4h
        .word   09188h
        .word   03331h
        .word   06662h
        .word   0CCC4h
        .word   089A9h
        .word   0373h
        .word   06E6h
        .word   0DCCh
        .word   01B98h
rtw_len .set 16
        .sect   "table"         ; LTU (based on bytes)
t_start
        .word   00h
        .word   01021h
        .word   02042h
        .word   03063h
……
        .word   0ED1h
        .word   01EF0h
t_len   .set    256
```

Annexe B - Exemples de modélisations Matlab

Dans cette annexe, nous présentons des exemples de modélisation des erreurs aléatoires, ainsi qu'en rafale, sur lesquels nous avons vérifié que les simulations permettent de vérifier les caractéristiques principales des codes CRC. Nous présentons également une modélisation possible de la technique de la fragmentation.

Avant de représenter les différents types d'erreurs qui peuvent altérer les messages, nous montrons, à la figure B.1, le multiplexage TDMA des messages codés (avec la génération des bits de contrôle de CRC). Pour cela et pour simplifier la modélisation, nous utilisons deux blocs de génération aléatoire de données, suivant une *loi de Bernoulli*, représentant deux nœuds du système de communication. Dans cet exemple, ces blocs génèrent des messages de 100 bits, codés avec le bloc *CRC Generator*, puis émis chacun dans leur fenêtre temporelle correspondante, à l'aide des blocs *Matrix Concatenation* et *Matrix Interleaver*. Ces messages sont alors placés, avant d'être émis, dans le bloc *Rx_Signal*.

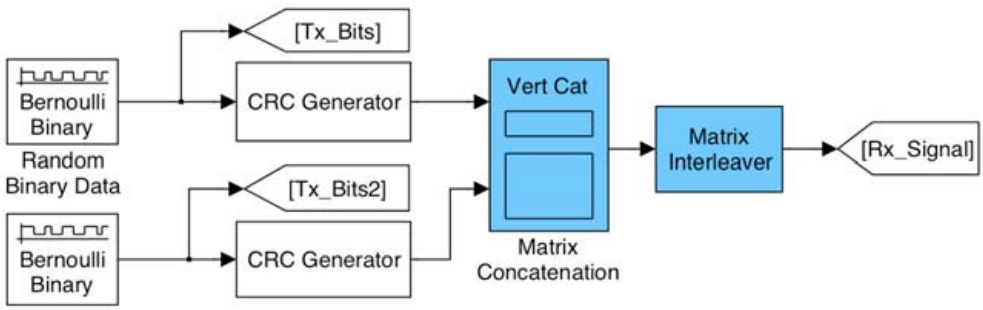


Figure B.1 - Génération des messages codés avec multiplexage TDMA

B.1 Modélisation des erreurs aléatoires

Les erreurs aléatoires peuvent être modélisées de deux façons différentes en utilisant soit le bloc *BSC*, soit le bloc *Binary Vector Noise Generator*.

(a) Modélisation sur la base du bloc BSC

Les messages du bloc *Rx_Signal* sont transmis sur un canal BSC (*Binary Symetric Channel*) dont on peut modifier le bruit en agissant sur la valeur du BER (Bit Error Rate). Ensuite, ces messages sont démultiplexés avec le bloc *Multiport Selector*.

Du côté récepteur, nous générons, à l'aide du bloc *CRC Syndrome Detector*, le champ de contrôle d'erreur *CRC Syndrome* (cf. figure B.2). Ce champ représente une valeur booléenne qui vaut 1 s'il n'y a pas d'erreur détectée, et 0 sinon. Disposant ainsi de ce champ de contrôle pour chacun des messages reçus, nous pouvons déterminer le taux de messages erronés non détectés avec la méthode présentée dans la section 4.2.3.3.b.

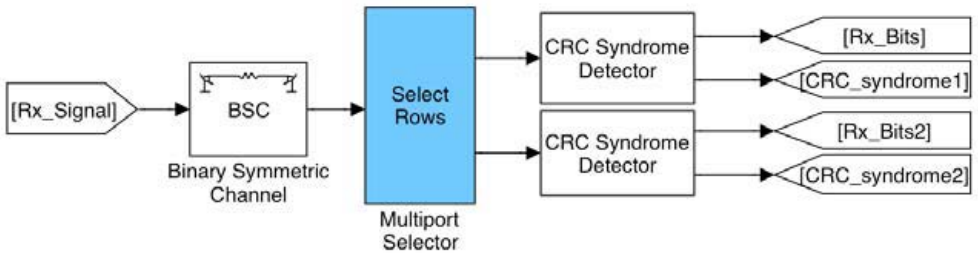


Figure B.2 - Transmission des messages codés via un canal BSC et détection d'erreurs

À partir de ce taux, il est nous est alors possible de tracer la courbe de P_{ue} (Probabilité de non détection des erreurs de transmission) en fonction du BER du canal de transmission BSC.

Les résultats des simulations effectuées sur plusieurs codes CRC standard donnent des valeurs de P_{ue} qui sont proches des valeurs théoriques, calculées avec la formule détaillée dans le chapitre 2. À titre d'exemple, en faisant des simulations pour un BER de 0,1 et sur 5.10^5 messages émis, de taille 100 bits et codés avec un CRC standard 16 bits (CRC-CCITT), le nombre de messages erronés non détectés est égal à 6, donnant une probabilité de non détection de $1{,}2.10^{-5}$, soit le même ordre de grandeur que la valeur théorique qui vaut 2^{-16} ou $1{,}525.10^{-5}$.

(b) Modélisation sur la base du bloc "Binary Vector Noise Generator"

La figure B3 montre le modèle que nous avons construit pour introduire des erreurs aléatoires, autrement que par la fonctionnalité proposée par le bloc BSC. Ces erreurs sont générées ici par le bloc *Binary Vector Noise Generator* sous forme d'un vecteur dans lequel on peut définir le nombre et la position des 1 qui viendront s'ajouter (avec l'opérateur logique XOR) au message codé issu du bloc *Rx_Signal*. Le message ainsi généré est erroné, et le nombre et la position des erreurs aléatoires qu'il comprend correspondent respectivement au nombre de 1 et à leur position dans le vecteur engendré suivant une loi de Bernoulli. Dans l'exemple ci-dessous, nous utilisons deux sources de génération aléatoire d'erreurs, représentées par les deux blocs *Binary Vector Noise Generator*.

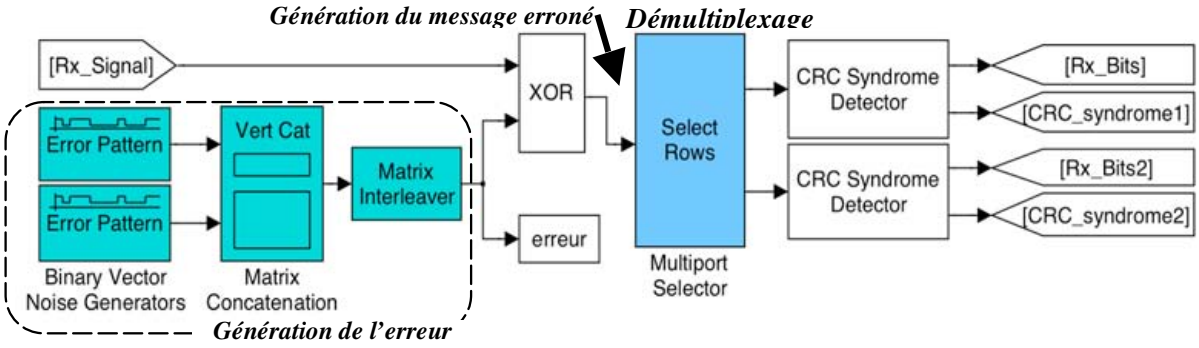


Figure B.3 - Génération d'un vecteur d'erreurs aléatoires

Après avoir testé l'intégrité des messages erronés avec le bloc *CRC_Syndrome_Detector*, qui génère le *CRC_Syndrome* relatif à chaque message, nous pouvons calculer la probabilité de non détection des erreurs de transmission et donc tracer la courbe qui représente son évolution en fonction du nombre d'erreurs introduites par message (cf. figure B.4).

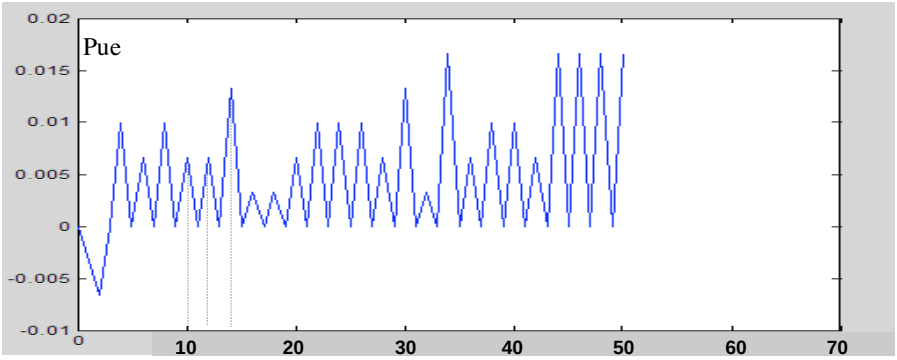


Figure B.4 - Probabilité de non détection en fonction du nombre d'erreurs

La courbe P_{ue} en fonction du nombre d'erreurs montre que les erreurs simples et doubles sont détectées ainsi que toutes les erreurs en nombre impair. Cela vérifie l'une des principales caractéristiques concernant le pouvoir de détection des codes CRC. Les valeurs négatives de P_{ue} sont dues au fait que des erreurs sur le champ de contrôle d'erreur sont détectées même si le message initial (sans le champ de contrôle d'erreur du CRC) ne contient pas d'erreur.

B.2 Modélisations des erreurs en rafale

Par rapport au modèle précédent, seule la représentation et la génération de l'erreur change. Le vecteur d'erreurs, qui vient s'ajouter au message émis, est créé cette fois à partir d'un script Matlab et intégré au modèle Simulink dans le bloc *errors* (cf. figure B.5).

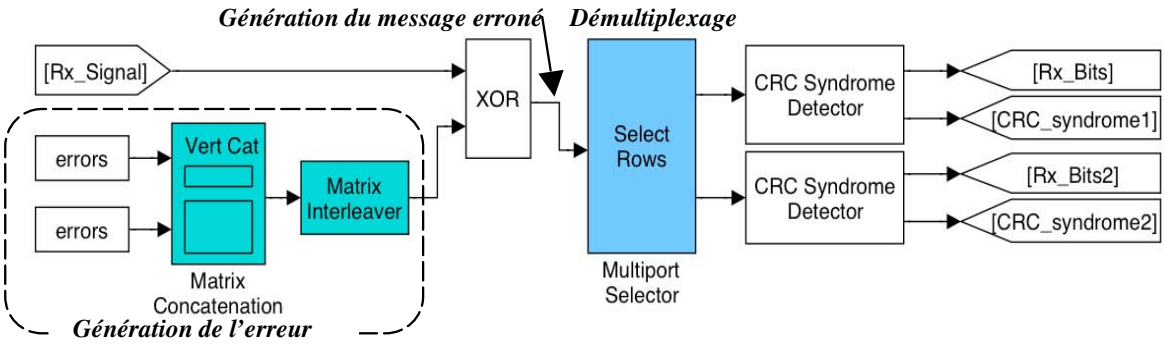


Figure B.5 - Génération d'erreurs en rafale

Pour simuler des erreurs en rafale (*burst*) et exécuter la simulation pour différentes longueurs (en bits) du *burst*, nous avons écrit un script en Matlab qui permet de créer un vecteur d'erreurs dans lequel des blocs de *i* "1" (*i* variant entre 1 et 15, et "1" représente une inversion de bit) apparaissent à des intervalles aléatoires. La distance entre deux blocs de "1" consécutifs est un entier aléatoire entre 1 et 40. L'erreur est ensuite récupérée dans le modèle avec le bloc *errors*.

La courbe de P_{ue} en fonction de la longueur du champ des erreurs en rafale (*burst*) pour un CRC 16 bits est nulle pour tous les *bursts* de longueur inférieure à 16. Ce qui vérifie une autre caractéristique des pouvoirs de détection des codes CRC.

B.3 Modélisation du principe de fragmentation

Nous modélisons dans cette section la technique de fragmentation présentée dans le chapitre 3 (cf. § 3.2.2). Celle-ci peut être modélisée dans Matlab-Simulink en spécifiant un nombre de champs de contrôle supérieur à 1 en paramètre du bloc *CRC Generator* (ou du bloc *CRC-N Generator*) et en utilisant des blocs *Submatrix* qui permettent d'extraire *m* sous-matrices à partir d'une matrice (M, N).

En partant d'un message de 100 bits générés aléatoirement suivant la loi de Bernoulli, on représente la matrice d'entrée (*M*=1, *N*=100) du bloc *Submatrix* correspondante à cette série de bits. Ensuite, ce bloc est paramétré de façon à ce qu'il génère deux sous-matrices (*M*=1, N=50) représentant respectivement les 50 premiers bits du message initial et les 50 derniers. Enfin, nous appliquons à ces deux messages un code CRC à 8 bits, et on transmet les deux messages codés sur un canal BSC avec un multiplexage TDMA. On aboutit ainsi au modèle de la figure B.6.

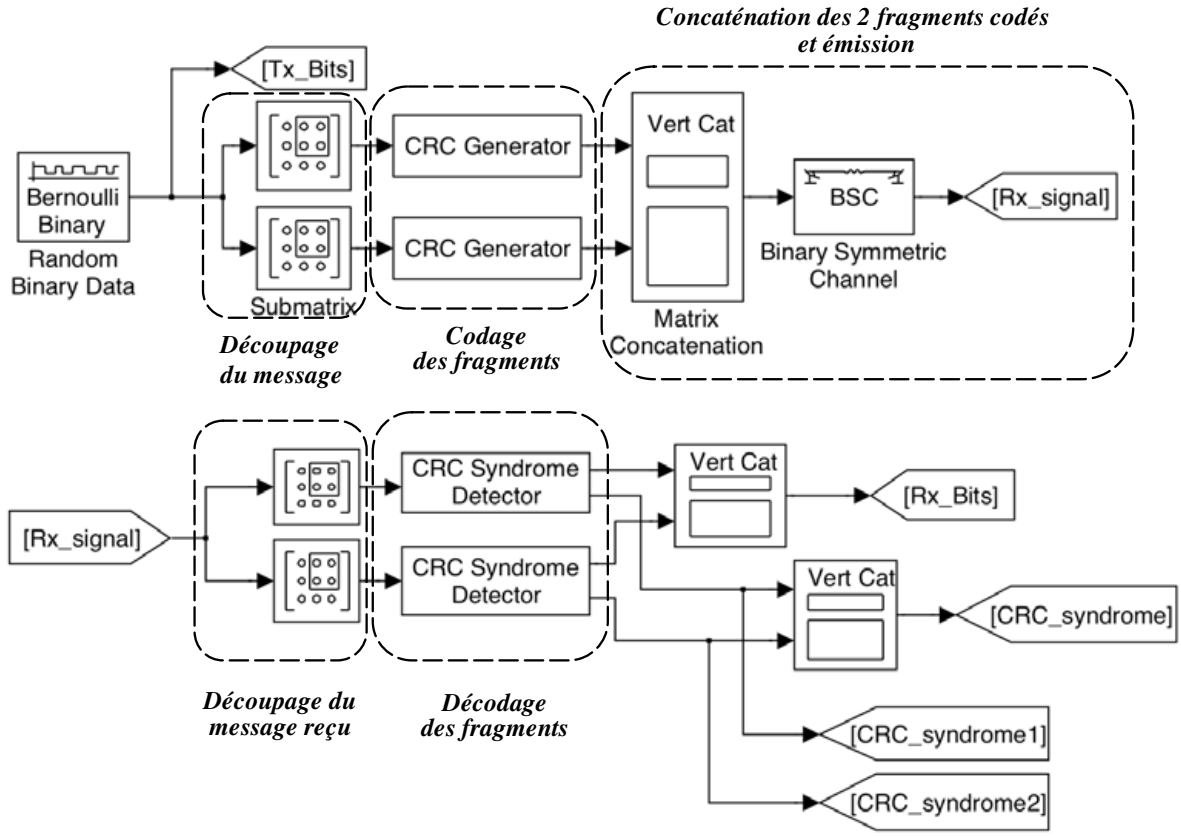


Figure B.6 - Mise en œuvre de la technique de fragmentation